TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

TERO MEHTÄNEN

OHJELMISTOROBOTIIKAN KEHITTÄMINEN JA KÄYTTÖÖNOTTO KETTERILLÄ MENETELMILLÄ

Diplomityö

Tarkastaja: professori Samuli Pekkola
Tarkastaja ja aihe hyväksytty
24. syyskuuta 2018

TIIVISTELMÄ

TERO MEHTÄNEN: Ohjelmistorobottien kehittäminen ja käyttöönotto ketterillä menetelmillä
Tampereen teknillinen yliopisto
Diplomityö, 48 sivua
Marraskuu 2018
Johtamisen ja tietotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Tuotantotalous
Tarkastaja: professori Samuli Pekkola

Avainsanat: ohjelmistorobotiikka, ohjelmistotekniikka, ketterät menetelmät

Tämän tutkimuksen tarkoitus oli luoda pohja ohjelmistorobotiikan osaamiselle Patrian tietohallinnossa. Tutkimuksen tavoitteena oli selvittää, miten ketteriä menetelmiä voidaan soveltaa ohjelmistorobottien kehittämisessä ja käyttöönotossa sekä mitä silloin tulee yrityksessä ottaa huomioon. Tutkimus koostuu neljästä osiosta, kirjallisuusselvityksestä, tapaustutkimuksista, tutkimustuloksien pohdinnasta sekä lopuksi yhteenvedosta.

Tutkimus aloitettiin esittelemällä tietojärjestelmätutkimuksen viitekehys, jota oli tarkoitus soveltaa Patriassa. Tämän jälkeen tehty kirjallisuusselvitys aloitettiin esittelemällä ohjelmistorobotiikka ja sen soveltamiseen liittyviä näkökulmia sekä määriteltiin keskeisiä kriteereitä ohjelmistorobotiikalla automatisoitaville prosesseille. Lisäksi kirjallisuusselvityksessä käytiin läpi ohjelmistokehitysprosessi, esiteltiin ketterät menetelmät sekä vertailtiin niitä perinteisiin ohjelmistokehitysmenetelmiin, luoden perusteet ketterien menetelmien soveltamiselle Patrian tapaustutkimuksissa.

Tutkimustuloksina esiin nousi ketterien menetelmien keskeisimmät periaatteet Patrian viitekehyksessä. Tapaustutkimusten perusteella vuorovaikutus asiakkaan kanssa, muutoksiin reagointikyky sekä ohjelmistorobottien ohjelmistokehityksen iteratiivinen luonne nousivat keskeisimmiksi tekijöiksi, joihin tulee kiinnittää erityistä huomiota jatkossa. Lisäksi työn myötä käsitys ohjelmistorobotiikan roolista organisaatiossa tarkentui ja siihen liittyvä viitekehys esiteltiin.

ABSTRACT

TERO MEHTÄNEN: Developing and implementing robotic process automation with agile methods
Tampere University of Technology
Master of Science Thesis, 48 pages
November 2018
Master's Degree Programme in Management and Information Technology
Major: Industrial Management
Examiner: Professor Samuli Pekkola

Keywords: robotic process automation, software engineering, agile methods

The purpose of this study was to provide the foundation for the center of competence of robotic process automation in the Patria department of information management. The objective of the study was to clarify how agile methods can be adapted in developing and implementing of robotic process automation and what must be taken into consideration. The study consists of four parts; literature report, case studies, reasoning of results and summary at the end.

The study was begun by presenting the frame of reference of the information system research which was implemented in Patria's environment. After that, a literature report was introduced containing fundamentals of robotic process automation and essential criteria for the processes to be automated. Then software development process was introduced, and agile methods were presented, and they were compared with traditional software development model creating the grounds to the adapting of agile methods in the case studies of Patria.

As results of this study few main principles of agile methods emerged. The most crucial aspects in developing and implementing robotic process automation are interaction with the customer on daily basis, ability to adapt with rapid rate of requirement changes and importance of noticing iterative characteristic of robotic process automation software development in Patria's environment. Furthermore, the role of the robotic process automation in the organization became clearer with this study and framework considering to that was created and introduced.

ALKUSANAT

Tähän diplomityöhön päättyy vuonna 2011 itselleni asettama tavoite valmistua jonain päivänä Tampereen Teknillisestä Yliopistosta diplomi-insinööriksi. Perimmäisenä syynä oli pyrkimys päästä niin vaativampiin kuin monipuolisempiinkin työtehtäviin. Tavoite tuntui tuolloin erittäin kaukaiselta ajatukselta, mutta luottamus sen toteutumiseen on ollut koko ajan läsnä. Matka on tähän mennessä ollut paljon antoisampi kuin ehkä osasin kuvitella.

Suuri kiitos mahdollisuudesta tehdä tämä työ kuuluu Patrialle. Oli loistava tilaisuus tehdä diplomityö työsuhteessa. Erityiskiitos kuuluu esimiehelleni, Vesa Rönkölle, jonka kanssa käydyt keskustelut toivat tähän diplomityöhön paljon hyviä näkökulmia.

Lisäksi haluan kiittää professori Samuli Pekkolaa toimivasta yhteistyöstä sekä asiallisesta diplomityön ohjaamisesta.

Tampereella, 13.11.2018

Tero Mehtänen

SISÄLLYSLUETTELO

1. JOHDANTO

Tämä diplomityö on tehty Patria-konsernin tietohallintoon. Patria-konserni muodostuu Patria Oyj emoyhtiöstä sekä sen omistamista tytäryhtiöistä. Patria-konserni jakautuu viiteen eri liiketoimintayksikköön, joita ovat Aviation, Systems, Land, Aerostructures ja International Support Partnerships. Näiden lisäksi Patria omistaa 61,8% Millog Oy:stä, joka vastaa Suomen puolustusvoimien materiaalihuollosta sekä sen kaluston huoltamisesta ja kunnossapidosta. Ampumatarvikkeita valmistavasta yhtiöstä nimeltä Nammo Lapua Oy, Patria omistaa puolet. Koko Patria-konsernin omistus on puolestaan jaettu Suomen hallituksen (50,1%) sekä norjalaisen puolustustarvikkeisiin keskittyvän yrityksen nimeltään Kongsberg Defence Systems (49,9%) kesken [16].

Patrian tarjoama vaihtelee huomattavasti sen eri liiketoimintojen välillä aina panssaroiduista ajoneuvoista ja kranaatinheitinjärjestelmistä siviililentäjäkoulutukseen sekä poliisihelikoptereiden kunnossapitoon. Tammikuussa 2017 Patrian hallitus hyväksyi uuden strategian sekä esitteli sen henkilöstölleen. Tämän uuden strategian mukaan Patria keskittyy jatkossakin varmistamaan ilmailu- ja sotilasasiakkailleen heidän kalustonsa käytettävyyden sekä tarjoamaan niiden suorituskyvyn jatkuvaa kehittämistä. Trendien mukaisesti kyseisessä strategiassa uutena erityiskysymyksenä on digitalisaatio sekä sen mukanaan tuomat mahdollisuudet kehittää organisaation toimintaa sekä ennen kaikkea tuottaa lisäarvoa asiakkaille [16].

Digitalisaatio on yksi aikamme megatrendeistä. Sen on sanottu muokkaavan maailmaa jopa enemmän kuin teollinen vallankumous. Yhdessä globalisaation kanssa digitalisaatio on muovannut toimialojen arvoverkkoja, hämärtänyt toimialojen rajoja ja tuonut markkinoille aivan uudenlaisia toimijoita. Muutokset markkinoilla ovat mahdollisuuksia, mutta eivät takeita markkinoilla menestymisestä. On selvää, ettei mikään toimija jää digitalisaation vaikutuksien ulkopuolelle vaan yritykset joutuvat ennemmin tai myöhemmin huomioimaan toiminnassaan digitalisaation vaikutukset [4]. Digitalisaation myötä ohjelmistoja ja kaikenlaisia digitaalisuutta hyödyntäviä sovelluksia on ympärillämme jatkuvasti enemmän. Tämä kehitys luo yrityksille aivan uudenlaisia mahdollisuuksia kehittää ja muokata liiketoimintaansa. Uudet mahdollisuudet tuovat mukanaan myös uusia uhkia, mikäli organisaatiot eivät kykene tunnistamaan liiketoimintaansa vaikuttavia tekijöitä ja trendejä.

1.1 Tutkimuksen tausta

Erilaisista liiketoiminnoista koostuvassa Patria-konsernissa on käytössä monia eri tietojärjestelmiä. Kuten tavallista, tiedon täytyy liikkua eri tietojärjestelmien välillä. Tavallisesti tietoa siirretään esimerkiksi toiminnanohjausjärjestelmien välillä automaattisesti, eli integroimalla kyseiset järjestelmät keskenään. Vaikka järjestelmiä päivitetään ja kehitetään jatkuvasti, nykyisin käytössä olevat tietojärjestelmät eivät ole vielä niin automatisoituja, että esimerkiksi yrityksen kirjanpitoon liittyvät prosessit tapahtuisivat automaattisesti, vaan kyseiset prosessit vaativat edelleen huomattavia määriä ihmisten tekemää manuaalista työtä. Lisäksi on syytä huomata, että yritysten liiketoimintojen suunnalta tulevat muuttuvat vaatimukset vaikuttavat oleellisesti tietojärjestelmiltä vaadittuja ominaisuuksien kehittymiseen. Mikäli käytössä olevien tietojärjestelmien toisiinsa integroiminen tai jälkikäteen niiden ominaisuuksien muuttaminen ei ole järkevää esimerkiksi taloudellisista syistä, voidaan ohjelmistorobotiikka nähdä yhtenä vaihtoehtona lisätä tietojärjestelmien automaatiotasoa. Näin ollen, Patria on ryhtynyt tutkimaan, miten ohjelmistorobotiikkaa hyödyntämällä sen sisäisiä prosesseja voidaan tehostaa ja nostaa niiden automaatiotasoa.

1.2 Tutkimusongelma

Tämän tutkimuksen on tarkoitus tuottaa tietoa ohjelmistorobotiikan soveltamisesta Patrian tietohallinnossa, taloushallinnollisten prosessien automatisointiin. Tutkimuksessa keskitytään menetelmiin, joilla ohjelmistorobotteja kehitetään sekä käyttöönotetaan. Työtä lähdettiin viemään eteenpäin hyvin vähäisillä lähtötiedoilla ja näin olleen tutkimuskysymyksenkin muotoutuminen vei aikansa, mutta lopulta yhteinen näkemys löydettiin ja tutkimuksen tavoitteet tunnistettiin. Tämän jälkeen varsinainen tutkimustyö voitiin aloittaa.

Tutkimuskysymys:

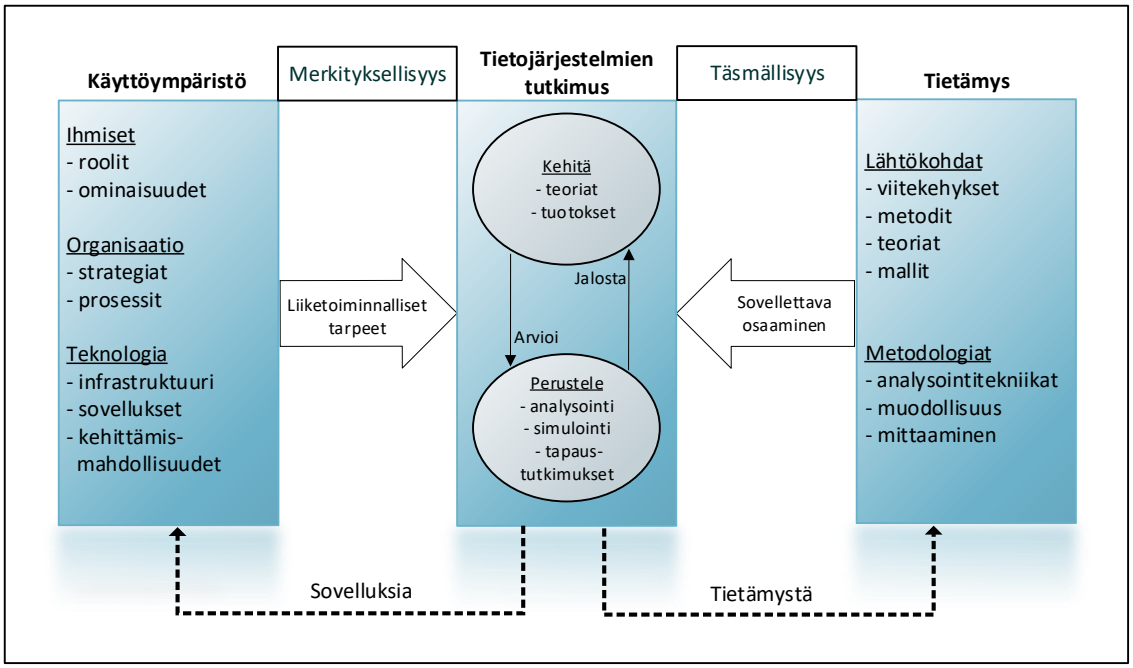
- **Miten ohjelmistorobotteja kehitetään ja käyttöönotetaan ketterillä menetelmillä?**

Tutkimuksen tavoitteena on luoda pohja ohjelmistorobotiikan osaamiselle Patriassa. Patrian kannalta merkittävän työstä tekee se, että aihe on uusi, eikä selkeää kuvaa ohjelmistorobotiikan hyödyntämisestä tällä hetkellä ole. Lisäksi jo alkuvaiheessa oli selvää, että potentiaalisia automatisoitavia prosesseja olisi runsaasti. Todettiin, että näiden prosessien tehokkaaseen automatisoimiseen tarvitaan vakioitu ohjelmistorobottien kehittämis- ja käyttöönottoprosessi, joten tämä tutkimus pyrkii vastaamaan siihen. Tutkimuskysymykseen vastataan arvioimalla ketterien menetelmien yhteensopivuutta Patrian toimintaympäristöön. Lisäksi tutkimuksessa tuodaan esille muita sen myötä esiin nousevia ohjelmistorobotiikkaan ja sen soveltamisympäristöön vaikuttavia seikkoja, jotka on syytä ottaa huomioon.

1.3 Tutkimusmenetelmä

Tutkimus tehtiin ohjelmistokehitysprosessista, jolla ohjelmistorobotteja kehitetään ja käyttöönotetaan. Tarkoituksena oli tutkia ketterien menetelmien soveltuvuutta kyseiseen prosessiin. Tutkimus aloitettiin kirjallisuusselvityksellä, jonka perusteella luotiin kriteerit ohjelmistorobotiikalla automatisoitaville prosesseille. Tämän jälkeen etsittiin tutkimustietoa ketteristä menetelmistä ja niiden soveltuvuudesta ohjelmistokehitykseen. Aineiston hakuun käytettiin Tampereen Teknillisen Yliopiston kirjastoa, Andor-portaalia ja Google Scholar-hakukonetta. Tutkimusmenetelmänä käytettiin tapaustutkimusta, joka on erään määritelmänsä mukaan empiirinen tutkimus, jossa jotain ajankohtaista ilmiötä sovelletaan käytännössä, eikä näiden välinen yhteys ole täysin ilmeinen [27]. Tutkimuskysymyksen muodostumisen jälkeen tunnistettiin tietojärjestelmien tutkimuksen viitekehys sekä sen keskeiset elementit (Kuva 1.). Tietojärjestelmien tutkimus on yhdistelmä niin käyttäytymistieteen kuin järjestelmäsuunnittelun paradigmoja. Tämä tarkoittaa, sitä ettei tietojärjestelmien tutkimus rajaudu vain kyseessä olevan järjestelmän tutkimukseen vaan myös käyttöympäristön, jossa sitä sovelletaan.

Sovellettaessa tietojärjestelmien tutkimuksen viitekehystä Patrian ympäristöön tässä tutkimuksessa tarkoitetaan käyttöympäristöllä taloushallintoa ja tietämyksellä taas tarkoitetaan tietohallintoa. Taloushallinnossa on syntynyt tarpeita hyödyntää digitalisaation mukanaan tuomia mahdollisuuksia eli tässä tapauksessa ohjelmistorobotiikkaa. Tietohallinto taas pyrkii vastaamaan näihin tarpeisiin tarjoamalla asiantuntijoidensa osaamista ja menetelmiä. Malleja tai menetelmiä ohjelmistorobotiikan soveltamiseen ei Patriassa kuitenkaan ole ollut, joten niitä ryhdyttiin kehittämään.



Kuva 1. *Tietojärjestelmien tutkimuksen viitekehys, perustuen lähteeseen [6].*

Kuten kuvasta (Kuva 1.) huomataan, on tietojärjestelmän tutkimuksen keskiössä tuotoksen tai teorian kehittäminen analysoimalla, simuloimalla tai toteuttamalla tapaustutkimuksia. Tässä tapauksessa prosessin kehittämiseen sovelletaan ketteriä menetelmiä, kuten aiemmin mainittiin ja näiden soveltuvuutta Patrian viitekehykseen arvioidaan tapaustutkimusten perusteella. Tämän prosessin myötä taloushallinto saa sovelluksia ja tietohallinto puolestaan lisää tietämystä aiheesta.

1.4 Tutkimuksen rakenne

Työ on jaettu kirjallisuusselvitykseen (luvut 2 ja 3), soveltavaan osioon (luku 4), tapaustutkimuksiin (luku 5), pohdintaan (luku 6) sekä yhteenvetoon (luku 7). Luvussa kaksi käsitellään prosessien automatisointia, esitellään ohjelmistorobotiikka ja tarkastellaan siihen liittyviä keskeisiä näkökulmia, kuten hyötyjä, haittoja ja riskejä. Lisäksi kyseisessä luvussa luodaan pohja automatisoitavien prosessien arvioinnille, joka on yksi keskeisin lähtökohta ohjelmistorobotiikassa. Kolmannessa luvussa käydään läpi keskeisiä ohjelmistokehityksen elementtejä niin perinteisillä menetelmillä kuin ketterillä menetelmilläkin. Luvussa neljä ohjelmistorobotteja tarkastellaan Patrian viitekehyksessä ja luvussa viisi toteutetaan kaksi tapaustutkimusta. Tapaustutkimuksissa saatua tietämystä ohjelmistorobotiikan soveltamisesta hyödynnetään seuraavassa luvussa, jossa esitellään tutkimustulokset ja pohditaan niiden merkitystä.

Tutkimuksen lopuksi tehdään yhteenveto ja käydään läpi mahdollisia jatkotutkimusehdotuksia sekä tulevaisuuden suuntaviivoja ohjelmistorobotiikan viitekehyksessä.

2. AUTOMATISOINTI

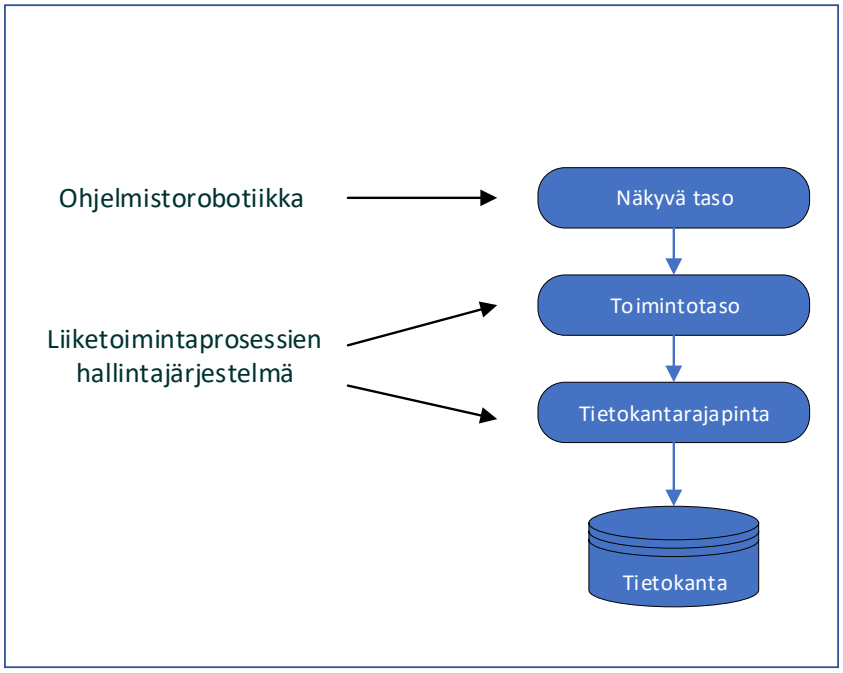
Tässä luvussa esitellään prosessien automatisointi sekä käydään läpi mitä ohjelmistorobotiikka ylipäätään tarkoittaa. Automaatio on ollut läsnä valmistavassa teollisuudessa jo pitkään. Esimerkiksi nykyajan autotehtaat ovat täynnä autoja valmistavia robotteja. Tämä kehityskulku ei enää rajoitu vain valmistavaan teollisuuteen vaan on saavuttanut myös toimistot sekä tietotyön. Toimistoissa ihmiset tekevät edelleen paljon rutiininomaisia tehtäviä, aivan kuten ennen tehtiin autotehtaan tuotantolinjallakin. Kuten tuotantolinjalla toimivan käsivarsirobotin ja toimistossa toimivan ohjelmistorobotin toimintaperiaatteet ovat samanlaisia: Ne pyrkivät jäljittelemään ihmisen toimintaa rutiininomaisissa tehtävissä.

2.1 Ohjelmistorobotiikka

Ohjelmistorobotiikka on käsitteenä hieman harhaanjohtava, koska sanan loppuosasta ”robotti” mieleen tulee fyysinen robotti. Ohjelmistorobotiikasta käytetään usein lyhennettä (RPA), joka tulee englannin kielen sanoista Robotic Process Automation. Kyse ei kuitenkaan ole robotista siinä mielessä kuin se yleisesti ymmärretään, vaan tietokoneella suoritettavasta ohjelmakoodista, joka jäljittelee ihmisen toimintaa, kuten esimerkiksi täyttää lomakkeita, kirjaa niitä tietojärjestelmään tai tarkistaa tietoja. Kyseessä ei ole järjestelmäintegraatio, vaan ohjelmistorobotti toimii aivan kuten järjestelmän käyttäjä toimisi. Järjestelmäintegraatiolla tarkoitetaan ohjelmistojen tai tietojärjestelmien yhteen liittämistä niin, että ne kykenevät vaihtamaan tietoa keskenään. Ohjelmistorobotti sen sijaan toimii tietojärjestelmässä, kuten esimerkiksi yrityksen toiminnanohjausjärjestelmässä käyttäjärajapinnan kautta, jolloin sitä on mahdollista soveltaa hyvinkin erilaisten tietojärjestelmien kanssa [1, 2].

Ohjelmistorobotiikka ei varsinaisesti ole osa yrityksen tietoteknistä infrastruktuuria vaan se toimii kyseisen infrastruktuurin päällä, kuten alla olevasta kuvasta (Kuva 2.) voidaan todeta. Muutamat ominaisuudet erottavat ohjelmistorobotiikkateknologian liiketoimintaprosessien hallintajärjestelmästä (Business Process Management System, BPMS). Liiketoimintaprosessien hallintajärjestelmällä pyritään nimensä mukaisesti koordinoimaan liiketoimintaan liittyviä prosesseja. Toisin kuin BPMS:ssä ohjelmistorobotiikkasovellukset eivät vaadi ohjelmointitaitoja ohjelmistojen rajapintojen konfiguroimiseksi vaan tämä tapahtuu ”drag and drop” – periaatteella, jossa toiminnallisia lohkoja lisätään projektiin kursorilla vetämällä. Toinen merkittävä ero näiden kahden lähestymistavan välillä on se, ettei ohjelmistorobotiikassa varsinaisesti luoda uutta sovellusta, joten tietomalleja ei tarvitse tehdä, kuten BPMS -ratkaisuissa. Näiden tekijöiden lisäksi on syytä huomata, että ohjelmistorobotti ei tallenna tietoa suoraan tietokantaan vaan se tapahtuu kuvassa (Kuva 2.) esitettyjen kerrosten kautta. Liiketoimintaprosessien hallintajärjestelmät tuleekin

nähdä toisiaan täydentävinä työkaluina hallita yrityksen tiedonkäsittelyyn liittyviä prosesseja [5, 17].



Kuva 2. *Ohjelmistorobotiikan rajapinta, perustuen lähteeseen [5].*

Puhuttaessa ohjelmistorobotiikasta tarkoitetaan varsin rutiininomaisten tehtävien automatisointia. Tällä hetkellä ohjelmistorobotiikkaa voidaan soveltaa vain tietyn tyyppisissä tehtävissä. Näiden tehtävien tulee olla selkeästi määriteltyjä, sääntöihin perustuvia, eivätkä ne saa sisältää ihmismäistä päättelyä [1]. Jonkin asteista ihmiselle tyypillistä päättelykykyä edellyttävien prosessien automatisointiin sovelletaan kognitiivista automaatiota. Tällaisen järjestelmän tulee kyetä tulkitsemaan järjestäytymätöntä tietoa ja tekemään sen pohjalta johtopäätöksiä, jotka esitetään yleensä todennäköisyyksinä. Ohjelmistorobotiikka on tämän vastakohta, koska sen antama ulostulo on deterministinen [17].

Sovellusteknologiat, joilla ohjelmistorobotteja kehitetään ja käytetään ovat lisenssipohjaisia. Capgeminin [11] mukaan ohjelmistorobottilisenssi maksaa noin yhdestä kolmasosasta yhteen viidesosaan täysipäiväisen työntekijän palkkaamisen kustannuksesta. Lacityn ja Willcocksin [12] mukaan yksi ohjelmistorobotti kykenee suorittamaan ennaltamääriteltyjä, rakenteellisia tehtäviä yhtä paljon kuin kahdesta viiteen ihmistä. Näistä lupaavista tuloksista huolimatta ohjelmistorobotit eivät kuitenkaan sovellu kaikkien liiketoimintaprosessien automatisointiin.

2.2 Soveltuvuuden arviointi

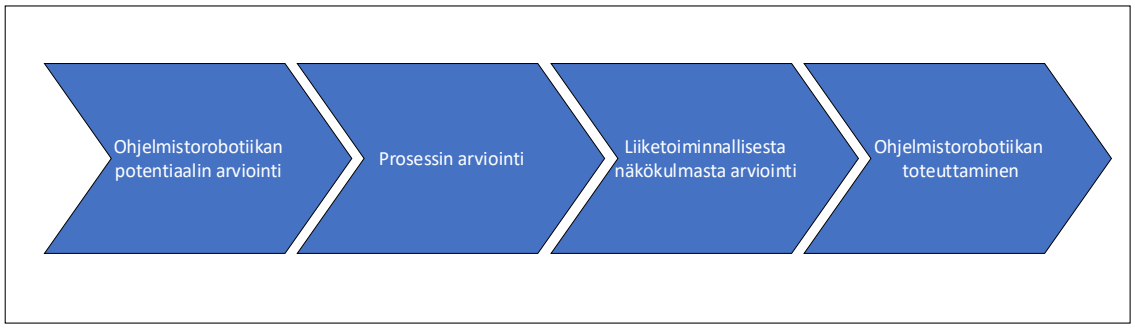
Ohjelmistorobotiikan hyödyntämisessä yksi keskeinen näkökulma ovat prosessit, joita sillä on tarkoitus automatisoida. Automatisoinnissa ei ole mitään uutta, sillä yritykset ovat pyrkineet automatisoimaan prosessejaan jo sitä pitkään. Automatisoitavan prosessin tunteminen ja sen ymmärtäminen ovat ensiarvoisen tärkeitä asioita, sillä jos prosessia ei tunneta riittävän hyvin sen automatisoiminen saattaa aiheuttaa enemmän haittaa kuin hyötyä. Kun prosesseja automatisoidaan käyttäen jotain työkalua, kuten ohjelmistorobotiikkaa kannattaa prosessia yleensä kehittää samalla [32]. On syytä muistaa, että vaikka prosessi todettaisiin mahdollisesti automatisoivaksi ei sen automatisoiminen välttämättä ole liiketoiminnallisessa mielessä kannattavaa.

Käsitteenä prosessi on erittäin laaja ja sillä voidaan tarkoittaa lähes mitä tahansa tapahtumaketjua. Tässä tutkimuksessa prosessia käsitellään ainoastaan pintapuolisesti esittelemällä, mitä se tarkoittaa. Prosesseiksi kutsutaan toiminta- ja tapahtumaketjuja, joista voidaan muodostaa loogisia kokonaisuuksia. Organisaatioissa on tyypillisesti monenlaisia prosesseja. Näihin prosesseihin vaikuttavat tietyt luonnonlait, joten ne tunnistamalla ymmärretään myös paremmin organisaation toimintaa ja voidaan tehostaa sitä. Organisaatioiden toimintaa tarkastelemalla prosessi usein ymmärretään asiakkaalle lisäarvoa tuottavana toimintana, johon käytetään resursseja. Projektin kertaluontoisuudesta poiketen prosessille tyypillistä on sen usein toistuva luonne. Prosessit voidaan jakaa kahteen päätyyppiin, ydinprosesseihin ja tukiprosesseihin. Ydinprosessit ovat kytköksissä suoraan asiakkaaseen. Tyypillisiä ydinprosesseja ovat tuotteiden ja palveluiden kehittäminen, tilauksista sopiminen sekä asiakastuki. Tukiprosessit puolestaan ovat yrityksen sisäisiä prosesseja, joiden tarkoitus on tukea ydinprosesseja. Tukiprosesseilla on yrityksen sisäisiä asiakkaita ja toimittajia, joilla tarkoitetaan yrityksen toisia osastoja [35, 36].

Tässä tutkimuksessa läpi käytävät prosessit ovat yrityksen tukiprosesseja ja asiakas on samaan aikaan myös toimittaja. Kaikki yrityksen prosessit eivät kuitenkaan tuota lisäarvoa asiakkaalle. Kirjanpidon prosessit ovat yksi esimerkki tällaisista prosesseista. Siihen käytetään yrityksen resursseja, jonka kautta se aiheuttaa kustannuksia, mutta ei varsinaisesti tuota lisäarvoa yrityksen asiakkaalle. Kirjanpito on kuitenkin välttämätön tukiprosessi organisaation liiketoiminnalle.

2.3 Soveltuvuusarvioinnin vaiheet

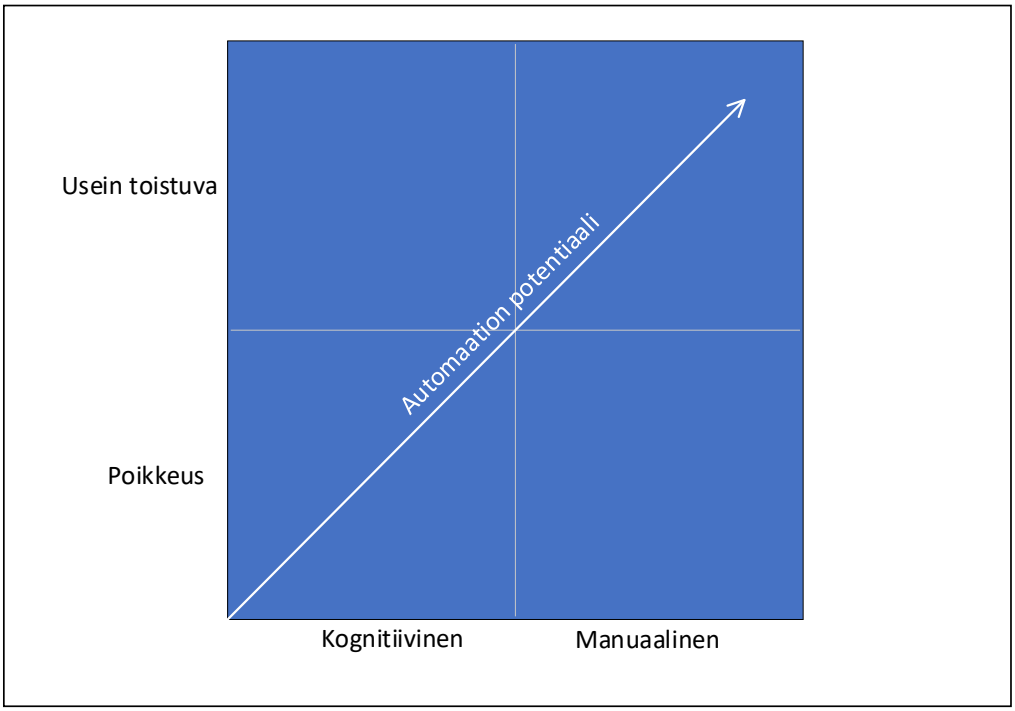
Ohjelmistorobotiikan soveltamiseen liittyy vaiheita, jotka tulee käydä läpi ennen kuin varsinainen ohjelmistorobotin kehitystyö aloitetaan. Oheisessa kuvassa (Kuva 3.) on esitelty kolme vaihetta, jotka toteuttamista edeltävät, Asatiani ja Penttinen [1] mukaan. Nämä toteutusta edeltävät vaiheet ovat tärkeitä, koska niiden avulla luodaan perusteet ohjelmistorobotiikan soveltamiselle [1].



Kuva 3. *Soveltuvuuden arviointi* [1].

Asatiani ja Penttinen [1] mukaan ensimmäisessä vaiheessa on tarkoitus ymmärtää ohjelmistorobotiikan potentiaali yrityksessä vielä varsin yleisellä tasolla. Tässä vaiheessa pyritään lisäämään organisaation ymmärrystä ohjelmistorobotiikasta sekä tunnistamaan potentiaalisia automatisoitavia prosesseja. Seuraavaksi ensimmäisessä vaiheessa tunnistettuja prosesseja ryhdytään arvioimaan tarkemmin. Tarkoituksena on ymmärtää kyseisten prosessien kulkua ja sekä niihin vaikuttavia sääntöjä. Lisäksi oleellista on pohtia, voidaanko automatisoitavaa prosessia muokata ohjelmistorobotille ”paremmin sopivaksi”, kuten esimerkiksi toteuttamalla jokin vaihe prosessista eri tavoin kuin aiemmin. Tämä vaihe kestää tavallisesti yhden päivän. Prosessien arvioinnin tueksi Fung [13] on määritellyt seuraavia keskeisiä kriteereitä prosessien arviointia varten ohjelmistorobotiikan näkökulmasta: [13]

1. Ei kognitiivisesti vaativia tehtäviä. Tehtävät eivät saa edellyttää ihmisille tyypillistä subjektiivista harkintaa, luovuutta tai päättelykykyä.
2. Korkea volyymi. Jotta ohjelmistorobotin kehittämistä, käyttöönottoa ja ylläpitoa kannattaisi tehdä tulisi tehtävän tulee toistua usein.
3. Ohjelmistorobotin hyödyntämistä kannattaa harkita, mikäli prosessi vaatii pääsyä useaan eri tietojärjestelmään. Tämä on ihmiselle huomattavasti työläämpää kuin ohjelmistorobotille.
4. Rajoittunut määrä poikkeuksia. Automatisoitavien tehtävien tulee olla pitkälle standardisoituja ja poikkeuksien määrän tulee olla rajallinen tai niitä ei saa olla ollenkaan.
5. Inhimilliset virheet. Mikäli tehtävässä esiintyy ihmisen tekemiä inhimillisiä virheitä, voidaan niitä pyrkiä välttämään automatisoinnilla.
6. Vakaa tietojärjestelmä. Tietojärjestelmän, jota ohjelmistorobotti käyttää tulisi pysyä suhteellisen vakaana sekä muuttumattomana, sillä ohjelmistorobotiikan perustuessa käyttöliittymän automatisointiin se ei todennäköisesti kykene suorittamaan ohjelmia, mikäli tietojärjestelmää muutetaan.
7. Ymmärrystä kustannusrakenteesta. Yrityksen on tärkeä ymmärtää prosessiin liittyvä kustannusten muodostuminen, jotta ohjelmistorobotin hyödyntämistä voidaan arvioida liiketoiminnallisista näkökulmista perustellusti.



Kuva 4. *Automatisoinnin potentiaali [1].*

Yllä olevassa kuvassa (Kuva 4.) on esitetty automaation potentiaali suhteessa prosessin luonteeseen. Edellä mainittuja kriteerejä sekä kuvaa (Kuva 4.) tarkastelemalla vahvoja ehdokkaita ohjelmistorobotiikan soveltamiskohteiksi ovat manuaaliset, usein toistuvat pitkälle standardisoidut tehtävät, joissa poikkeuksien lukumäärä on rajallinen.

Teknologinen kehitys on mahdollistanut aina vain monimutkaisempien tehtävien automatisoinnin, eikä tämä kehitys näytä hidastuvan. Jon-Arild Johannessen [24] mainitsee neljännen teollisen vallankumouksen, jonka keskeisinä ajureina ovat toimineet globalisaatio ja digitalisaatio. Tällä vallankumouksella tarkoitetaan älykkyydellä varustettuja tuotteisiin integroituja robotteja, joita voidaan ohjelmoida aistimaan ympäristöään, ymmärtämään mitä tapahtuu sekä muokkaamaan toimintaansa palautteen perusteella niin, ettei niitä tarvitse ohjelmoida uudelleen [24]. Tarkasteltaessa automatisoinnin potentiaalia (Kuva 4.) tämä kehitys tarkoittaa siirtymää lähemmäs kohti nuolen alkupistettä. Toisin sanoen tulevaisuudessa potentiaalisesti automatisoitavien tehtävien määrä kasvaa teknologisen kehityksen myötä.

Voidaan todeta, että tekniset valmiudet toteuttaa ohjelmistorobotteja paranevat muun teknologisen kehityksen myötä, mutta tämä ei vielä yksin riitä. Ohjelmistorobottien toteuttamisen tulee olla perustelua myös liiketoiminnallisessa mielessä. Soveltuvuuden arvioinnin kolmannessa vaiheessa (Kuva 3.) edellä tunnistettuja prosesseja arvioidaan liiketoiminnallisesta näkökulmasta. Tässä vaiheessa varsinaisesti todetaan ohjelmistorobotin tuottama hyöty yritykselle tarkastelemalla ohjelmistorobottien ja ihmisten välistä kustannustehokkuutta sekä tuottavuutta [1]. Mikäli ohjelmistorobotin hyödyntäminen kyseisen

prosessin osalta todetaan vielä tässäkin vaiheessa kannattavaksi ratkaisuksi, siirrytään viimeiseen vaiheeseen, jossa varsinainen ohjelmistokehitys aloitetaan. Tähän palataan tarkemmin seuraavassa pääluvussa.

2.4 Hyödyt ja haitat

Kuten jo aiemmin mainittiin, toimii ohjelmistorobotti tietojärjestelmässä pääosin kuten ihmiskäyttäjä, tästä syystä ohjelmistorobottien käyttöönotto ei edellytä muutoksia sen käyttämään tietojärjestelmään. Tämä on itse asiassa yksi ohjelmistorobotiikan merkittävimpiä etuja, koska tietojärjestelmien ollessa monimutkaisia saattaa niiden integroinnista muodostua raskas ja kallis, jopa vuosien mittainen projekti. Tähän verrattuna ohjelmistorobotin suunnittelu, kehitys ja käyttöönotto vievät vain murto-osan tuosta ajasta. Ohjelmistorobotiikka nähdään kuitenkin vain väliaikaisena ratkaisuna, jolla pyritään täyttämään kuilu vanhojen manuaalisiin prosesseihin perustuvien tietojärjestelmien sekä uusien täysin automaattisesti toimivien tietojärjestelmien välillä [1].

Yritykset ovat tyypillisesti ulkoistaneet niitä toimintoja, jotka eivät ole sen ydinosaamista, kuten kirjanpito tai laskujen käsittely. Ulkoistamisen on todettu auttavan yrityksiä vähentämään henkilöstökustannuksia ja keskittymään ydinosaamiseensa, mutta haasteena ovat kasvavat hallinnolliset kulut sekä mahdollisesti monimutkaiset sopimukset alihankkijan ja asiakasyrityksen välillä [1].

Ohjelmistorobotiikan toinen merkittävä hyöty koskeekin ulkoistamista sen ollessa kustannustehokas ratkaisu arvoa tuottamattomien rutiinitehtävien, kuten kirjanpidon hoitamiseen. Väitetäänkin, että ihmiset, jotka tekevät rutiininomaisia työtehtäviä voidaan tulevaisuudessa siirtää tuottavampiin työtehtäviin ohjelmistorobotiikan soveltamisen myötä. Harrison ja Wanyama [33] osoittivat automatisoinnin lisänneen työntekijöiden kokemaa työtyytyväisyyttä tehden työstä aiempaa monipuolisempaa. Ainakin pitkällä aikavälillä työtehtävien luonne muuttuu automatisoinnin myötä suorittavasta työstä korkeampaa kognitiivista kykyä vaativiin tehtäviin, kuten hallinnollisiin tehtäviin, konsultointiin sekä data-analytiikkaan [1]. Vaikka ohjelmistorobotiikalla ei saavutettaisi merkittäviä liiketoiminnallisia hyötyjä, voi sillä olla suuri vaikutus työtyytyväisyyteen. Fassoulis ja Alexopoulos [34] puolestaan tutkivat työtyytyväisyyden vaikutuksia työntekijöiden tuottavuuteen. He totesivat työtyytyväisyyden ja tuottavuuden välisen korrelaation. Näin ollen ohjelmistorobotiikan kaikkien hyötyjen mittaaminen saattaa olla hankalaa sen välillisten vaikutusten takia.

Ohjelmistorobotiikalla saavutettavien hyötyjen ohella siihen liittyy luonnollisesti myös haittapuolia tai epäilyksiä, jotka koskevat yrityksen henkilöstöä. Tähän mennessä ohjelmistorobotiikan käyttöönotto yrityksissä on koettu positiiviseksi kehitykseksi, ilman merkittäviä työpaikkojen katoamisia. Kuten jo aiemmin mainittiin ohjelmistorobotit kykenevät suorittamaan ainoastaan hyvin määriteltyjä tehtäviä, joilla on selkeät säännöt. Tästä syystä sillä ei voi korvata tehtäviä, jotka edellyttävät ihmisen ongelmanratkaisuakykyä

tai päättelyä. Yritysten tulisinkin keskittyä kustannussäästöjen sijaan ohjelmistorobotiikan mukanaan tuomiin mahdollisuuksiin määritellä työtehtäviä uudelleen. Edellä mainituista seikoista huolimatta ohjelmistorobotiikkaan kohdistuu epäilyksiä, koska työntekijät saattavat silti kokea ohjelmistorobotit kilpailijoinaan. Tämä saattaa aiheuttaa jännitteitä esimiesten ja työntekijöiden välillä, sekä johtaa jopa työmoraalin heikkenemiseen työyhteisössä. Tästä syystä tavalla, jolla ohjelmistorobotiikasta yrityksessä keskustellaan sekä miten se henkilöstölle esitellään, on suuri merkitys työyhteisön kannalta [1, 28].

2.5 Riskit

Ohjelmistorobottien käyttöön ottaminen saattaa vaikuttaa petollisen yksinkertaiselta. Ohjelmistorobotiikan toimittajat luovat helposti kustannussäästöjä hakevalle yritykselle todellisuutta ruusuisemman kuvan ohjelmistorobotiikasta, sillä siihen liittyy myös riskejä, jotka on syytä tunnistaa. Yritykset usein aloittavatkin ohjelmistorobottihankkeensa kokeilemalla soveltaa sitä erilaisiin toimintoihin sekä tekemällä näistä johtopäätöksiä jatkoa varten. Ohjelmistorobotiikan soveltamiseen liittyy viisi keskeistä riskiä, jotka on hyvä tiedostaa yrityksessä [3].

Mikäli yrityksessä ei standardisoida ohjelmistorobotin kehitystä, saattaa siitä muodostua yritykseen uusi hankalasti hallittava tietojärjestelmä. Tästä syystä on tärkeä ymmärtää, että tehtäviä suorittavat ohjelmat ovat aivan samankaltaisia ohjelmakoodeja kuin muissakin sovelluksissa ja niihin täytyy suhtautua samalla tavalla. Lähtökohtana tulee olla ohjelmakoodien uudelleen käytettävyys, ohjelmointi riittävän abstraktilla tasolla, versionhallinta sekä lokitiedoston kirjoittaminen. Näin ollen ohjelmistorobotin soveltamisen tulee olla tiukasti koordinoitua toimintaa liiketoimintaan osallistuvien henkilöiden sekä teknologiatiimin välillä. Lisäksi ohjelmistorobotit tulisi testata samalla tavoin kuin muutkin yrityksessä tuotantoon tulevat ohjelmistot [3].

Ohjelmistorobotit saattavat tehdä niiden käyttämien tietojärjestelmien päivittämisestä hitaampaa ja vaikeampaa. Koska ohjelmistorobotti käyttää alla olevia tietojärjestelmiä, kuten Windowsia, on se riippuvainen siitä ja muutokset kyseisessä järjestelmässä saattavat aiheuttaa ongelmia ohjelmistorobotin toiminnassa. Toisin kuin ihminen ohjelmistorobotti ei sopeudu muutoksiin samalla tavalla, ja jopa pienet muutokset esimerkiksi käyttöliittymän ulkoasussa saattavat johtaa ohjelmistorobotin toimimaan virheellisesti. Mikäli ohjelmistorobotin uudelleen ohjelmointi tai päivittäminen on kovin työlästä, nostaa se kynnystä päivittää taustalla olevaa tietojärjestelmää [3].

Ohjelmistorobottien laaja soveltaminen liian nopeasti saattaa vaarantaa siinä onnistumisen. Alkuvaiheessa liian laajan lähestymistavan ottamisen riskinä on, että se aiheuttaa merkittäviä kuluja ennen kuin yritys on edes päättänyt miten se ohjelmistorobotti investointeja hyödyntää tehokkaasti. Parempi strategia on aloittaa ohjelmistorobotiikan hyödyntäminen pienin askelin, osoittaa ratkaisun toimivuus ja sitten laajentaa muihin kohteisiin [3]. Tietämyksen lisääntyessä yrityksen on helpompi arvioida ja mitata potentiaalisia automatisoitavia prosesseja ja niiden vaikutusta esimerkiksi kustannusnäkökulmasta.

On väärin olettaa prosessien omistajien olevan oikeita henkilöitä automatisoimaan prosessinsa. Toisaalta prosessien omistajien täytyy olla mukana projektissa, koska vain heillä on tarvittava osaaminen, jotta prosessin jokainen vaihe voidaan ohjelmoida ohjelmistorobotille. Lisäksi prosessin omistajalla on paljon sellaista taustatietoa, jota tarvitaan ohjelmistorobotin kehitystyössä. Järkevä ratkaisu voisi olla prosessien omistajien kanssa läpi käytävä potentiaalisten automatisoitavien prosessien tarkastelu ja sitten ulkopuolisen toteutuksesta vastaavan henkilön mukaan tuominen [3].

Kun organisaatiot rakentavat ohjelmistorobotiikkaa koskevia strategioita sekä taktisia suunnitelmia on syytä huomata, etteivät ohjelmistorobotit poista organisaation tarvetta jatkuvasti uudelleen arvioida tietojärjestelmiään. Tästä hyvänä esimerkkinä kannattaa pitää mielessä ”vasara ja naula” -analogia. ”Kun annat jollekin uuden kiiltävän vasaran, yhtäkkiä jokainen ongelma näyttää naulalta.” On totta, että ohjelmistorobotiikkaratkaisut auttavat automatisoimaan manuaalisia prosesseja sekä parantamaan tuottavuutta, mutta on olemassa myös muita työkaluja, joilla voidaan saavuttaa jopa parempi tuottavuus ja kustannussäästöt. Nämä työkalut pitävät sisällään niin prosessin alusta loppuun digitalisointia, nopeaa prosessin uudelleen suunnittelua kuin koneoppimistakin [3]. Eli kuten ”vasara ja naula” -analogiasta voidaan päätellä, ohjelmistorobotiikka ei ole ainoa työkalu pyrittäessä tuottavuuden parantamiseen tai kustannussäästöjen saavuttamiseen.

Useissa organisaatioissa teknologinen infrastruktuuri on kärsinyt investointien puutteesta. On houkutteleva mutta harhaanjohtava ajatus, että ohjelmistorobotiikka korvaisi puutteet tietojärjestelmissä. Ohjelmistorobotiikkaa soveltavan yrityksen on tästä uudesta työkalusta huolimatta silti aina uudelleen arvioitava ja pohdittava kuinka heidän tietojärjestelmiään tulisi modernisoida. Riskinä on, että muutaman onnistuneen ohjelmistorobotiikalla toteutetun sovelluksen jälkeen yrityksen johto päättää pyrkiä välttelemään kustannuksia, jolloin investointeja organisaation teknologiseen infrastruktuuriin ei tehdä, koska ajatellaan ohjelmistorobottien korvaavan ne. Asia ei kuitenkaan ole näin, vaan lähtökohtana tulee olla liiketoimintaa tukeva infrastruktuuri [3].

3. OHJELMISTOKEHITYSMENETELMÄT

Tässä luvussa keskitytään tarkastelemaan ohjelmistokehitysmenetelmiä. Luvun alussa esitellään yleisesti ohjelmistokehitysprosessi, jonka jälkeen käydään läpi erilaisia lähestymistapoja koordinoida tätä prosessia, niin ketterin menetelmin, kuin perinteisin menetelmin. Luvussa esitellään nämä erilaiset ketterät menetelmät, mutta niitä ei tarkastella sen syvällisemmin. Erilaiset ketterät menetelmät pitävät sisällään useita erilaisia menetelmiä ja käytäntöjä, joilla on yhteisiä tekijöitä, kuten iteratiivisuus. Työn kannalta oleellista on ketterien menetelmien filosofian ymmärtäminen ja niiden erottaminen perinteisistä menetelmistä.

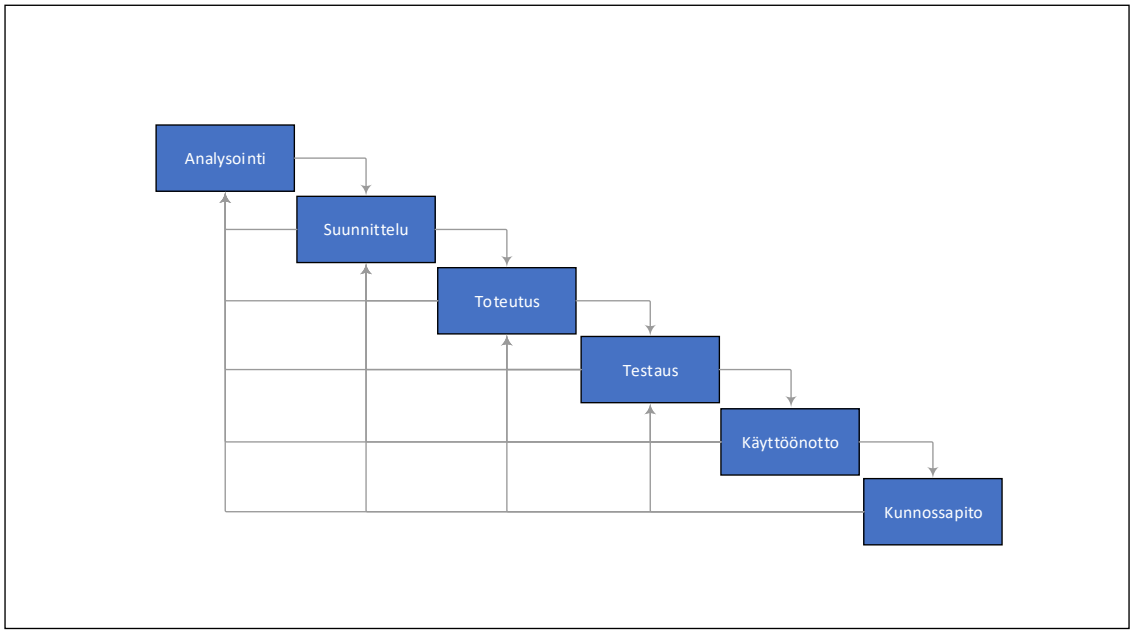
3.1 Ohjelmistokehitysprosessi

Ohjelmistokehitysprosessista puhuttaessa tarkoitetaan prosessia, jolla ohjelmistoa suunnitellaan, kehitetään ja lopuksi käyttöönotetaan [19]. Ohjelmistokehityksen keskeisenä tavoitteena on luoda laadukkaita ohjelmistoja sekä tehokkaasti että tuottavasti. Jotta tähän tavoitteeseen on mahdollista päästä, tarvitaan erilaisia lähestymistapoja, projektista ja asiakastarpeista riippuen [20]. Ohjelmistokehitystä voidaan tehdä erilaisilla menetelmillä. Pääpiirteissään nämä menetelmät jakautuvat kahteen luokkaan, joita ovat ketterät menetelmät sekä perinteisempi menetelmä nimeltään vesiputousmalli. Menetelmillä on luonnollisesti erilaisia ominaisuuksia, joten ne sopivat erilaisiin projekteihin omine vahvuuksine ja heikkouksine. Menetelmän soveltuvuuteen tiettyyn projektiin vaikuttaa oleellisesti mm. kyseisen projektin koko, tuoteprofiili, kilpailutilanne sekä itse asiakas [18]. Ohjelmistokehityksen tuottavuus ja lopputuotteen laatu ovat kytköksissä itse ohjelmistokehitysprosessin laatuun. Näin ollen kehittämällä ohjelmistokehitysprosessia voidaan parantaa myös edellä mainittujen tekijöiden tasoa [21]. Tästä syystä ohjelmistokehitysmenetelmän onnistunut valitseminen on oleellista.

3.2 Vesiputousmalli

Perinteisissä ohjelmistokehitysmalleissa, kuten vesiputousmallissa (Kuva 6.) ohjelmistokehityksen ajatellaan tapahtuvan vakaissa ja systemaattisissa jaksoissa, vaiheittain [10]. Esitystavasta riippuen vesiputousmallissa on viidestä kuuteen vaihetta, jotka on tarkoitus toteuttaa järjestyksessä niin, että seuraava vaihe voidaan aloittaa vasta kun edellinen vaihe on saatu päätökseen. Bassil [19] toteaa, että menetelmän haasteina ovat olleet huomattavat budjetin ylitykset, viivästyneet toimitukset sekä tyytymättömät asiakkaat. Pääsyinä edellä mainituille haasteille voidaan pitää resurssien allokointia virheellistä menetelmän eri vaiheiden välille. Mikäli projektille osoitettujen resurssien allokointi projektin läpiviennissä epäonnistuu, ilmenee tämä toisissa vesiputousmallin mukaisissa vaiheissa heik-

kona resurssien käyttötehokkuutena sekä vastaavasti toisissa vaiheissa resurssit osoittautuvat riittämättömiksi. Tämä saattaa johtaa pullonkaulojen muodostumiseen sekä projektin läpimenoajan pidentymiseen ja tätä kautta kasvaviin kustannuksiin [19].



Kuva 5. *Vesiputousmallin vaiheet, perustuen lähteeseen [18].*

Yllä olevassa kuvassa (Kuva 5.) on esitelty vesiputousmallille tyypilliset vaiheet. Ensimmäisessä vaiheessa eli analysointivaiheessa määritellään ja analysoidaan järjestelmän vaatimukset perusteellisesti. Tähän vaiheeseen sisältyy järjestelmän tarkastelu niin liiketoiminnallisesta näkökulmasta kuin itse järjestelmän näkökulmasta. Tyypillisesti toiminnalliset vaatimukset määritellään erilaisten mahdollisten tapausten perusteella, jossa käyttäjä on vuorovaikutuksessa järjestelmän kanssa. On sanottu, että tämä vaihe on yksi ohjelmistokehitysprojektin haasteellisimpia vaiheita, koska tässä vaiheessa pitäisi osata ottaa huomioon käytännössä kaikki järjestelmään liittyvät seikat [10, 19].

Ensimmäisessä vaiheessa tehdyn analyysin sekä järjestelmän vaatimusten perusteella tulisi järjestelmästä olla kattavasti tietoa, jotta sen varsinainen suunnittelu voidaan aloittaa. Suunnitteluvaiheessa määritellään suunnitelma, jolla järjestelmä voidaan rakentaa, kuten millaisia algoritmeja tarvitaan, millainen on järjestelmäarkkitehtuuri, tietokantatyyppi tai miltä käyttäjärajapinta näyttää. Tämän jälkeen toteutusvaihe voidaan aloittaa. Nimensä mukaisesti tässä vaiheessa luodaan vaatimusten mukainen järjestelmä. Toteutusvaihetta seuraa testausvaihe, jossa testataan, täyttääkö järjestelmä sille asetut vaatimukset. Kuten minkä tahansa testaaminen saadaan tämän vaiheen perusteella palaute projektin aiempien vaiheiden suoriutumiselle. Kun järjestelmä on todettu toimivaksi ja vaatimusten mukaiseksi, se otetaan käyttöön. Käyttöönottovaihetta seuraa järjestelmän ylläpito eli käytännössä asiakastuki. Ylläpitovaihe pitää sisällään järjestelmästä mahdollisesti myöhemmin löytyvien virheiden korjaamista ja järjestelmäpäivitysten tarjoamista asiakkaalle [23].

Elämme alati muuttuvassa maailmassa, eikä liiketoiminnan tarpeet joihin ohjelmistoa suunnitellaan sekä toteutetaan pysy aina samana, vaan ne saattavat muuttua nopeasti. Vesiputousmallin heikkous on juuri tässä. Siinä vaiheessa, kun asiakkaalle tarjotaan alussa määriteltyjen vaatimusten mukaista järjestelmää saattaa olla niin, että asiakkaan tarpeet ovat jo muuttuneet huomattavasti. Vesiputousmallin mukaan tehtävässä ohjelmistokehityksessä muutosten tekeminen myöhäisessä vaiheessa saattaa olla työlästä, koska alussa suoritetussa vaatimusten määrittelyssä niitä ei välttämättä ole osattu ottaa huomioon. Ketterät menetelmät pyrkivät paikkaamaan tätä vesiputousmallin heikkoutta nopealla muutoksiin reagoimisella [10, 18].

3.3 Ketterät menetelmät

Siitä kuinka ohjelmiston kehittäminen tulisi organisoida niin, että ohjelmistoratkaisujen toimittaminen olisi nopeaa, parempaa ja edullisempaa on käyty paljon keskustelua. Vuosien saatossa ratkaisuksi on ehdotettu monenlaisia ratkaisuja, aina erilaisista työkaluista, erilaisiin menetelmiin ja käytäntöihin. Viimeisin suuntaus pyrkiä parantamaan ohjelmistokehitysprosessia on nimeltään ketterät menetelmät [14]. Niistä on puhuttu yli vuosikymmenen ajan, mutta vieläkään siitä mitä ketteryys ohjelmistokehityksessä tarkoittaa, ei ole täysin yhteistä näkemystä. Kuten termistä voisi päätellä ohjelmistokehityksessä ketterät menetelmät tähtäävät nopeaan ohjelmiston kehittämiseen ympäristössä, jossa vaatimukset muuttuvat nopeasti [7]. Ketterät menetelmät ovat olleet kiistelty aihe ohjelmistokehitysyhteisössä. Toiset sanovat sen olevan parasta mitä ohjelmistokehitykselle on viime vuosina tapahtunut, kun taas toiset ovat täysin tätä näkemystä vastaan [23].

Keskeinen lähtökohta ketterälle ohjelmistokehitykselle on asiakastarpeen tyydyttäminen. Jotta tämä voidaan saavuttaa käytännöt kuten, kehitettävästä ohjelmistosta usein tapahtuvat versiojulkaisut ja asiakkaan kanssa aktiivinen vuorovaikuttaminen ovat ketterien menetelmien kulmakiviä. [9] Ketterät menetelmät voidaan nähdä pyrkimyksenä soveltaa ohjelmistokehitykseen alun perin 1950-luvun Japanista peräisin olevaa Toyotan käyttämänä tunnetuksi tullutta Lean-filosofiaa, jossa keskeistä on vähentää hukkaa, saavuttaa vaadittu laatu sekä keskittyä ongelman juurisyyn selvittämiseen [14]. Ericksson ja Lyytinen [15] määrittelevät ketteryyden seuraavasti:

"agility means to strip away as much of the heaviness, commonly associated with the traditional software-development methodologies, as possible to promote quick response to changing environments, changes in user requirements, accelerated project deadlines and the like."

Vuonna 2001 julkaistiin ketterän ohjelmistokehityksen julistus. Julistuksen mukaan ketterien menetelmien periaatteilla toteutettavassa ohjelmistokehityksessä on 12 keskeistä näkökulmaa [8]:

1. *”Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.”*
   Tärkeintä on täyttää asiakasvaatimukset tarjoamalla asiakkaalle ohjelmistoversioita aikaisessa vaiheessa sekä säännöllisesti. Jokainen ohjelmistoversio sisältää aiempaa versiota enemmän ominaisuuksia.

2. *“Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.”*
   Vaatimusten muuttaminen, jopa kehityksen myöhäisessä vaiheessa on hyväksyttävää. Ketterien menetelmien filosofiassa tämä nähdään positiivisena asiana, koska ymmärrys asiakkaan todellisista vaatimuksista kasvaa. Tämän myötä asiakastarve kyetään täyttämään paremmin.

3. *”Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.”*
   Toimivien ohjelmistoversioiden asiakkaalle tarjoaminen säännöllisesti, muutaman viikon tai muutaman kuukauden välein, jotta ohjelmistokehitystä voitaisiin ohjailla asiakkaan antaman palautteen pohjalta oikeaan suuntaan.

4. *”Business people and developers must work together daily throughout the project.”*
   Varsinaiseen liiketoimintaan osallistuvien henkilöiden sekä ohjelmistokehittäjien tulee työskennellä yhdessä päivittäin läpi projektin.

5. *“Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.”*
   Projektiin osallistuvat ihmiset ovat ketterissä menetelmissä erittäin keskeisessä roolissa ja heidän tulee olla motivoituneita. Heille tulee antaa sopiva ympäristö, tuki sekä luottaa siihen, että se saavat työn tehtyä.

6. *”The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.”*
   Tehokkaimpana tiedonvälityskanavana ketterään ohjelmistokehitykseen osallistuvan ryhmän välillä pidetään kasvotusten käytävää keskustelua. Toisin sanoen kaiken kattava dokumentointi ei ole ketterillä menetelmillä toteutettavan ohjelmistokehityksen keskeinen teema vaan menetelmät painottavat ihmisten välistä vuoropuhelua.

7. *”Working software is the primary measure of progress.”*
   Toimiva ohjelmisto on projektin ensisijainen etenemisen mitta. Toisin kuin perinteisissä menetelmissä ketterillä menetelmillä toteutettavassa ohjelmistokehityksessä etenemistä ei mitata, sillä kuinka monta vaihetta on saatu valmiiksi, vaan sillä kuinka paljon asiakkaan hyväksymiä toiminnallisuuksia on toteutettu.

8. *”Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.”*
   Tämä tarkoittaa sitä, että kehitystä tulee tehdä tiimille sopivalla vauhdilla, jotta vältyttäisiin ylitöiltä sekä työntekijöiden loppuun palamiselta. Tarkoitus on tehdä laadukasta kehitystyötä.

9. *”Continuous attention to technical excellence and good design enhances agility.”*
   Tekniseen suorituskykyyn ja hyvää suunnitteluun tulee kiinnittää huomiota ketteryyden korostamiseksi. Tällä tarkoitetaan, sitä että jos pyritään saavuttamaan korkealaatuinen lopputuote, tulee laatuun kiinnittää huomiota läpi kehitysprosessin.

10. *“Simplicity--the art of maximizing the amount of work not done--is essential.”*
    Ketterillä menetelmillä toteutettavassa ohjelmistokehityksessä tiimien tarkoitus ratkaista vain ne ongelmat, jotka ovat relevantteja projektin tavoitteiden kanssa. Mikäli asiakas haluaa lisätä tuotteeseen uuden ominaisuuden, yksinkertaisuus tarkoittaa, että asiakkaan pyyntöön voidaan helposti mukautua pitämällä suunnittelu mahdollisimman yksinkertaisena.

11. *“The best architectures, requirements, and designs emerge from self-organizing teams.”*
    Itseohjautuvalla ketterällä tiimillä on valtuudet päättää parhaista käytännöistä, kuten järjestelmäarkkitehtuurista, sen vaatimuksista sekä suunnittelusta. Vastuu projektin onnistumisesta on jokaisella tiimin jäsenellä.

12. *“At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.”*
    Säännöllisin väliajoin tiimi keskittyy pohtimaan mitkä ovat toimivia käytäntöjä ja mitkä eivät ole. Tämän perusteella tiimin jäsenet tarkastelevat omaa toimintaansa ja sitä, kuinka tulla tehokkaammiksi. Toisin sanoen omaa toimintaansa arvioimalla ketterä tiimi pysyy ketteränä.

Edellä esitellyistä ketterien menetelmien periaatteista voidaan tehdä yhteenveto neljään keskeiseen arvoon, joita ovat: [29]

- Yksilöt ja vuorovaikutus ennemmin kuin prosesseja ja työkaluja
- Toimiva ohjelmisto ennemmin kuin kaiken kattava dokumentointi
- Yhteistyö asiakkaan kanssa ennemmin kuin sopimusneuvottelut
- Muutoksiin reagointi ennemmin kuin suunnitelman mukaan eteneminen

Puhuttaessa ketteristä menetelmistä tarkoitetaan joukkoa erilaisia ohjelmistokehitysmenetelmiä. Näistä menetelmistä tunnetuimpia ovat: Extreme Programming, Scrum, Dynamic Systems Development Method (DSDM), Lean Development, Crystal ja Feature-Driven Development [23]. Nämä esitellään seuraavan sivun taulukossa (Taulukko 1.).

Taulukko 1. *Ketterät menetelmät.*

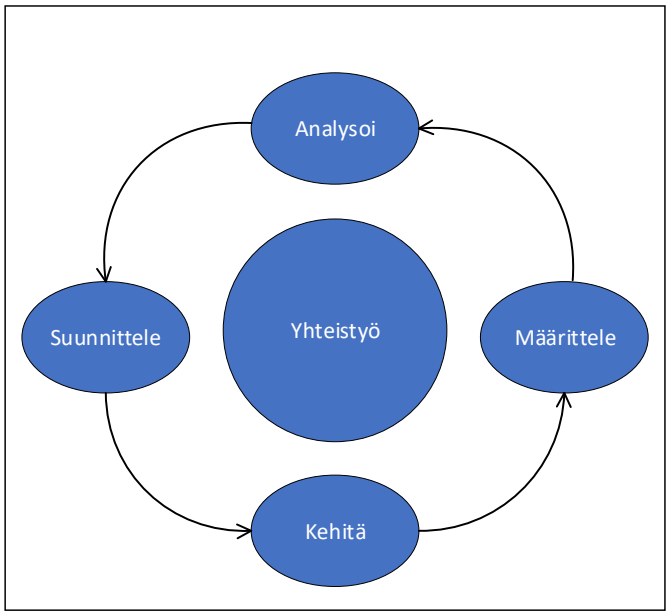
Menetelmä	Ominaisuudet
Extreme Programming	Extreme Programming on ohjelmistokehitysmenetelmä, jossa on neljä keskeistä arvoa. Nämä arvot ovat kommunikaatio, yksinkertaisuus, palaute sekä rohkeus kokeilla. Menetelmä pitää sisällään luonnollisesti myös neljä ohjelmistokehityksen perusominaisuutta kuten ohjelmoinnin, testaaminen ja debuggaamisen. [15]
Scrum	Scrum menetelmä on ohjelmistokehitysmenetelmä, joka on tarkoitettu pienille ryhmille, pieniin projekteihin, mutta se voidaan skaalata myös keskikokoisiin projekteihin. Ohjelmistokehitys tapahtuu iteratiivisesti sprinteissä, jotka kestävät kahdesta viikosta kuukauteen. Sprintillä toteutettavista toiminnallisuuksista ylläpidetään backlogia, eikä muutoksia kesken sprintin hyväksytä. [26]
Dynamic Systems Development Method	Dynamic Systems Development Method (DSDM) on enemmän viitekehys kuin menetelmä. Se on peräisin 1990-luvulta. Viitekehtys koostuu kuudesta vaiheesta, joita ovat esiprojekti, soveltuvuustutkimus, liiketoiminnallinen tutkimus, toiminnallisen mallin iterointi, suunnittelu ja kehitys, toteutus ja ylläpito. [23]
Lean Development	Lean Development on valmistavassa teollisuudessa paljon sovellettujen Lean-menetelmien tuomista ohjelmistokehitykseen. Keskeistä on hukan vähentäminen, asiakas tarpeen täyttäminen ja muutoksiin reagointi. [23]
Crystal	Alistar Cockburn kehitti Crystal-menetelmän 1990-luvulla. Lähtökohtana oli hänen näkemyksensä, jossa yksi ohjelmistokehityksen suurimpia haasteita oli vuorovaikutus. Crystal menetelmässä onkin keskeistä painottaa projektiin osallistuvien ihmisten välistä keskustelua, eikä muodollista dokumentointia. [23]
Feature-Driven Development	Menetelmä koostuu mallin kehittämisestä, ominaisuuksien listaamisesta ja niiden suunnittelusta, ominaisuuksien toteuttamisesta. Työhön osallistuvan ryhmän koko vaihtelee projektin vaativuuden mukaan. Yhden iteraatiokierroksen pituus on korkeintaan kaksi viikkoa. [23]

Näiden menetelmien (Taulukko 1.) keskeinen tavoite on luoda viitekehys ketterälle ohjelmistokehitykselle yrityksessä, mutta mitä se tarkoittaa? Kruchten [30] mukaan ketteryyttä ei pitäisi määritellä ainoastaan käytäntöjen mukaan vaan se tulisi nähdä laajemmin, organisaation kykynä reagoida ympäristössä tapahtuviin muutoksiin nopeammin kuin muutoksia tapahtuu. Kuten aiemmin todettiin, menetelmiä on erilaisia, mutta niillä on yhteisiä tekijöitä, kuten iteratiivinen sekä inkrementaalinen luonne. Ne kaikki jakavat ketterät menetelmät -julistuksen näkemykset ja arvot [23].

Alun perin vesiputousmallista kehitetyissä iteratiivisissa sekä inkrementaalisissa ohjelmistokehitysmalleissa projekti jaetaan vaiheisiin, joita toistetaan läpi kehitysprosessin.

Vesiputousmallista poiketen ketterissä menetelmissä tarkoituksena ei ole määritellä tarkasti koko järjestelmää yksityiskohtia myöten vaan jatkuvasti kehittää ja täsmentää niin vaatimuksia kuin tavoitteitakin [23]. Inkrementaalista mallia hyödyntämällä pyritään vähentämään ohjelmistokehitykseen kuluvaa aikaa jakamalla projektin vaiheisiin samalla tavalla kuin vesiputousmallissa, mutta toisin kuin vesiputousmallissa seuraavan vaiheen aloittaminen ennen kuin aiempi vaihe on saatu valmiiksi, on sallittua. Näin ollen säästyy aikaa, koska useampaa kuin yhtä vaihetta voidaan tehdä samanaikaisesti [23].

Iteratiivisessa mallissa projekti jaetaan eri mittaisiin sykleihin. Jokaisen sykli on kuin vesiputousmallin koko projekti, pitäen sisällään kaikki vaiheet aina vaatimusten analysoinnista toimittamiseen. Syklin lopputuloksena syntyy toimituskelpoinen tuote ja siihen liittyvä dokumentointi. Iteratiivisessa mallissa kehitys aloitetaan toteuttamalla järjestelmän yksinkertaisimmat ominaisuudet. Kehityksen jatkuessa, jokaisella kierroksella lisätään ominaisuuksia järjestelmään. Tällainen lähestymistapa mahdollistaa muutosten tekemisen myöhäisessäkin kehitysvaiheessa hyvin pienellä vaivalla [23].



Kuva 6. *Ketterien menetelmien iteratiivisuus, perustuen lähteeseen [8].*

Yllä olevassa kuvassa (Kuva 6.) on esitelty ketterille menetelmille tyypillinen niiden tyypillinen iteratiivinen luonne. Kuten aiemmin todettiin, jokaisella kierroksella järjestelmän ominaisuuksia kehitetään, jotta ne vastaavat paremmin asiakkaan vaatimuksia. Ketterät menetelmät painottavat erityisesti yhteistyötä asiakkaan kanssa, jossa vaatimusten muuttuminen kesken projektin nähdään normaalina osana kehitysprosessia. Toinen tyypillinen ero perinteisiin menetelmiin verrattuna on kehitykseen osallistuvien ihmisten jakautuminen pieniin itseohjautuviin ryhmiin, vahvasti ylhäältä johdetun organisaatiorakenteen sijaan [7].

3.4 Ketterien menetelmien hyödyt ja ongelmat

Kuten kaikilla menetelmillä niin myös ketterillä menetelmillä on niin hyvät kuin huonot puolensa. Dybån ja Dingsoyrin [14] ovat tutkineet ketterien menetelmien hyötyjä ja ongelmia. Heidän mukaan ketterien menetelmien hyviksi puoliksi on todettu palautteen saamisen aikaisessa vaiheessa sekä aktiivisempi vuorovaikuttaminen ohjelmistokehitykseen osallistuvien osapuolten välillä. Näiden seikkojen lisäksi säännölliset tapaamiset ovat auttaneet luomaan paremman kokonaiskuvan projektista ja sen etenemisestä. Lisäksi asiakkaan aktiivinen osallistuminen koettiin positiivisena asiana sekä asiakas koettiin erittäin tärkeänä tiedonlähteenä [31].

Dybån ja Dingsoyring [14] totesivat ketterien menetelmien ongelmiksi menetelmien heikon skaalautuvuuden projektin koon kasvaessa, pariohjelmointi nähtiin toisinaan uuvuttavana sekä ketterille menetelmille tyypillinen jatkuvan testaamisen koettiin vaativan suuria ponnisteluja. Aiemmin ketterien menetelmien hyödyiksi todettiin vuorovaikutuksen lisääntyminen, mutta vuorovaikutus tiimin ja muiden tiimien kanssa saattaa jäädä heikoksi. Näiden lisäksi tutkimuksessa [31] esiin nousi ketterän ohjelmistokehitystiimin johtamisen vaativan johtajalta riittävää koulutusta, koska menetelmät painottavat tiimien itseohjautuvuutta, mikä saattaa johtajalle olla hankalaa hyväksyä. On syytä huomata, että kyseisessä tutkimuksessa [31] analysoitiin lähinnä Extreme Programming menetelmällä toteutettuja tapauksia.

3.5 Yhteenveto eroavaisuuksista

Aiemmin tässä luvussa esiteltiin ketterät menetelmät ja perinteisistä menetelmistä vesiputousmalli. Näiden menetelmien välisiä eroja sivuttiin jo hiukan, mutta alla olevaan taulukkoon (Taulukko 2.) on koottu menetelmien eroavaisuuksia [18].

Taulukko 2. *Ohjelmistokehitysmenetelmien eroavaisuuksia, perustuen lähteeseen [18].*

	Perinteiset menetelmät	Ketterät menetelmät
Perusoletus	Järjestelmät ovat mahdollisimman tarkasti ennalta määriteltyjä ja niitä rakennetaan mukaillen tarkkaa ja ekstensiivistä suunnitelmaa	Korkealaatuinen vaatimuksiin mukautuva ohjelmisto, jota kehitetään pienissä ryhmissä käyttäen jatkuvan kehittämisen periaatteita. Perustuu nopeaan palautteen saamiseen ja muutosten hallintaan
Johtamistyyli	Käskeminen ja hallinta	Johtajuus ja yhteistyö
Tietämyksen hallinta	Täsmällinen	Epäsuora
Vuorovaikutus	Virallinen	Epävirallinen
Kehitysmalli	Elinkaarimalli (Vesiputousmalli)	Kehitys-toimitus malli
Haluttu organisaatiomuoto	Byrokraattinen	Joustava
Laadunhallinta	Tarkka suunnittelu ja tiukka hallinta. Myöhään tapahtuva raskas testaus	Jatkuva vaatimusten hallinta, suunnittelu ja testaus

Kuten aiemmin todettiin, onnistunut ohjelmistokehitysmenetelmän valinta on keskeisimpiä tekijöitä ohjelmistokehitysprojektin onnistumisen kannalta. Taulukkoa kaksi tarkastelemalla voidaan todeta perinteisten ohjelmistokehitysmenetelmien ja ketterien menetelmien olevan keskenään varsin erilaisia. Ne vaativat organisaatiolta hyvin erilaisia lähestymistapoja, kuten esimerkiksi johtamisen osalta. Luonnollisesti ohjelmistokehitysmenetelmän tulee sopia ympäristöön, jossa sitä sovelletaan. Ketterät menetelmät ovat saaneet kuitenkin alkunsa perinteisistä menetelmistä, mutta jatkuvasti muuttuvassa maailmassa on tarvittu joustavampaa lähestymistapaa kehittää ohjelmistoja. Tähän ketterät menetelmät pyrkivät. Vesiputousmalliin perustuva ohjelmistokehitys ei ole kadonnut mihinkään, mutta se on saanut rinnalleen uudenlaisia lähestymistapoja.

Käyttökohteet, joissa ohjelmistoja hyödynnetään vaihtelevat usein hyvinkin paljon riippuen toimialasta ja asiakastarpeista. Ohjelmistokehitysprosessin tulee kyetä tuottamaan vaatimukset täyttäviä ohjelmistoja asiakastarpeisiin, eikä ole vain yhtä tapaa organisoida ohjelmistokehitysprosessia. Tähän on kehitetty useita erilaisia lähestymistapoja. Toisaalla ohjelmistokehitysprojekti saattaa muodostua hyvinkin vaativaksi ja kalliiksi, mi-

käli projektin alussa ei kyetä määrittelemään ohjelmistolle asetettavia vaatimuksia riittävän tarkasti ja myöhemmin joudutaan toteuttamaan muutoksia. Toisaalta liian kevyt vaatimusten määrittely projektin alussa saattaa johtaa epäsuotuisiin lopputuloksiin, mikäli järjestelmä on monimutkainen. Vaihtelevista vaatimuksista ja erilaisista tarpeista johtuen, yhtä ainoaa ohjelmistokehitysmallia, joka kattaisi kaikenlaiset ohjelmistokehitysprojektit, ei voida määritellä. Tästä syystä ohjelmistokehityksessä on käytössä erilaisia malleja, joiden mukaan ohjelmistokehitystä voidaan jäsennellä ja johtaa, aina vesiputousmallista ketteriin menetelmiin.

4. OHJELMISTOROBOTIIKAN SOVELTAMINEN

Tässä luvussa keskitytään soveltamaan kirjallisuusosiossa esiteltyjä näkökulmia ja menetelmiä. Aiemmassa luvussa esiteltiin ketterät menetelmät ohjelmistokehityksessä. Kirjallisuusselvityksen mukaan ketterillä ohjelmistokehitysmenetelmillä on 12 keskeistä piirrettä, jotka käytiin läpi. Näistä piirteistä on tarkoitus löytää oleellisimmat näkökulmat Patrian viitekehykseen. Menetelmien soveltuvuuden arviointia suoritetaan tekemällä tapaustutkimuksia, joissa automatisoidaan jokin prosessi ohjelmistorobotiikalla. Lisäksi tapaustutkimuksien tuomaa tietämystä hyödynnetään ketteriin menetelmiin perustuvan ohjelmistokehitysprosessin arvioimisessa. Lopuksi esiin nostetaan tapaustutkimusten tulosten perusteella oleellisimmat Patrian ohjelmistorobotiikan soveltamisen kannalta tärkeimmät näkökulmat.

4.1 Tutkimusasetelma

Kuten monissa muissa yrityksissä niin Patriassakin ohjelmistorobotiikka on nähty digitalisaation ja teknologisen kehityksen mukanaan tuomana potentiaalisena vaihtoehtona parantaa yrityksen operatiivista tehokuutta automatisoimalla sen prosesseja. Ohjelmistorobotiikan soveltaminen yrityksissä on vielä uusi lähestymistapa kehittää sen prosesseja, ja näin ollen Patriassa haasteena onkin ollut ohjelmistorobotiikan osaamisen puute. Markkinoilla olisi tarjolla monia ohjelmistorobotiikkaa palvelunakin tarjoavia yrityksiä, mutta tällainen ratkaisu ei ole ollut vaihtoehto, koska toimiala, jolla Patriakin toimii, on erittäin tarkasti säädelty sekä valvottu. Patriassa on esimerkiksi paljon turvaluokiteltua tietoa, joka ei missään tapauksessa saa päätyä ulkopuolisten käsiin. Tästä syystä, jotta ohjelmistorobotiikkaa voitaisiin Patriassa ylipäätään hyödyntää, todettiin että alan osaamisen olisi oltava Patrialla itsellään. Edellä mainituista syistä syntyi tarve kehittää Patrian ohjelmistorobotiikan osaamista ja näistä lähtökohdista lähdettiin tekemään tätä tutkimusta sekä kehitystyötä ohjelmistorobotiikasta.

Aluksi ohjelmistorobottien ohjelmistokehitysprosessiksi oli määritelty vesiputousmallin mukainen lähestymistapa. Yritin aluksi soveltaa kyseistä lähestymistapaa ohjelmistorobottien kehittämiseen, mutta se ei tuntunut sopivan, koska vaatimukset muuttuivat nopeasti sekä ymmärrys ohjelmistorobotiikasta ei ollut riittävän tukevalla pohjalla. Minulta kesti jonkin aikaa havaita nämä haasteet ja ideoida uusi lähestymistapa.

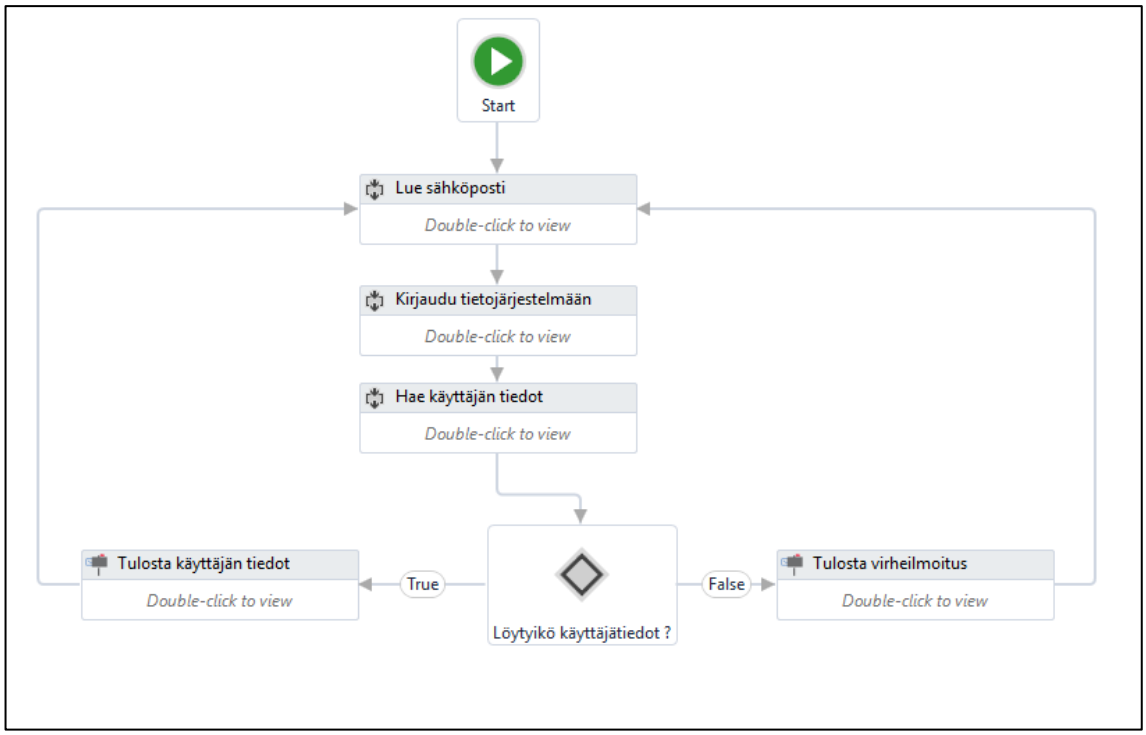
Ketterät menetelmät valikoituivat ohjelmistokehityksen lähtökohdaksi kahdesta syystä. Ensimmäinen syy oli se, että ohjelmistorobotit, joita ketterillä menetelmillä on tarkoitus toteuttaa eivät ole kovinkaan monimutkaisia tai laajoja järjestelmiä, jonka ansiosta vaatimusten määrittely ja järjestelmäsuunnittelu voitiin pitää kevyinä. Toinen syy valita ketterät menetelmät olivat jatkuvasti muuttuvat vaatimukset, joihin tuli kyetä vastaamaan

ohjelmistorobotteja kehitettäessä. Lisäksi on syytä huomata, että ohjelmistorobotiikka on Patrian viitekehyksessä täysin uusi menetelmä kehittää prosesseja, joten esimerkkitapauksia yrityksen sisältä ei ollut vaan ohjelmistorobotiikan soveltaminen alkoi alusta saakka.

4.2 Sovellusteknologiat

Projektin alussa käytiin läpi muutamia erilaisia vaihtoehtoja Patriassa käytettävälle ohjelmistorobotiikan sovellusteknologialle. Markkinoilla olevista ohjelmistorobotiikan sovellusteknologioista on pyritty tekemään sellaisia, että ihmiset jotka eivät osaa ohjelmoida, kykenisivät omaksumaan kyseisen sovellusteknologian nopeasti ilman todellista ohjelmointiosaamista. Huomasimme tämän aiheuttavan ongelmia, jos ohjelmistorobotille tarvitsee tehdä räätälöityä ohjelmakoodia, kuten jollain muulla ohjelmointikielellä toteutettuja toiminnallisuuksia. Näin ollen päädyimme sovellusteknologiaan nimeltään UiPath, joka nimensä mukaisesti soveltuu käyttäjärajapinnan kautta tapahtuvaan automatisointiin. BluePrism oli vahva kilpailija UiPath:lle, mutta sen käyttämisestä aiheutuvat lisenssikustannukset olivat huomattavasti UiPath:n lisenssikustannuksia korkeammat. Lisäksi UiPath tarjoaa kahden kuukauden mittaista ilmaista kokeilulisenssiä.

UiPath:ssa on kaksi osiota, UiPath Studio ja UiPath Robot. Studio on ohjelmistorobottien kehitystyökalu, kun taas Robot on ohjelmakoodien suorittamista varten. Johtuen aiemmasta ohjelmointikokemuksestani uusi ohjelmisto oli helppo omaksua, vaikka aluksi UiPath-käyttöliittymä tuntui vaikeaselkoiselta. Esimerkiksi If-rakenteiden toteuttaminen UiPath-sovellusteknologialla on haastavaa, koska käyttöliittymästä on pyritty tekemään niin visuaalinen kuin mahdollista, mikä taas johtaa siihen, että vähänkin ”syvempi” if-rakenne vaikeuttaa ohjelman luettavuutta.



Kuva 7. *UiPath-käyttöliittymä.*

Yllä olevassa kuvassa (Kuva 7.) on esimerkki UiPath-sovellusteknologialla toteutetusta ohjelmasta. Ohjelman suoritus alkaa sähköpostin lukemisella, jonka jälkeen ohjelmassa kirjaudutaan tietojärjestelmään ja etsitään käyttäjän tiedot ja tarkistetaan, löytyykö sähköpostissa olleen henkilön tiedot tietojärjestelmästä. Mikäli kyseisen käyttäjän tietoja ei löydy tietojärjestelmästä, tulostaa ohjelma virheilmoituksen ja siirtyy lukemaan uuden sähköpostin. Jokainen elementti sisältää sekvenssin, joka elementin aktivoituessa suoritetaan. UiPath-sovellusteknologialla ohjelmointi koostuu siis sekvensseistä ja virtauskaaviosta.

Valitsimme UiPath:n sovellusteknologiaksi, koska koimme että sillä pääsemme sujuvasti alkuun ja opimme nopeasti lisää ohjelmistorobotiikassa. Olemme kuitenkin päättäneet pitää myös avoinna mahdollisuudet käyttää tulevaisuudessa myös muita teknologioita ohjelmistorobottien kehittämiseen ja käyttämiseen. Esimerkiksi Python tukee testausautomaatiota, joka on hyvin pitkälti ohjelmistorobotiikan kaltaista osittaista käyttöliittymän automatisointia. Toinen alusta, jolla ohjelmistorobotteja Patriassa kehitetään, on Microsoft Visual Studio, jossa ohjelmoidaan C# ja Python ohjelmointikielillä. C# on mahdollista tehdä UiPath ympäristöön räätälöityjä ohjelmakoodeja, joilla voidaan esimerkiksi kutsua tietyn toiminnon suorittavaa funktiota. UiPath:ssa on myös jonkinlainen Python-ohjelmakoodien ajoon kykenevä laajennus, mutta sitä emme ole vielä kokeilleet. Tähän mennessä olemme käyttäneet Python-ohjelmointikielellä toteutettuja ohjelmakoodeja esimerkiksi XML-muotoisten tiedostojen käsittelyyn ja työkalujen, kuten tiedostojen siirtämiseen keskittyvien ohjelmakoodien luomiseen.

Lisäksi UiPath:ssa on syytä huomata kehitettävän ohjelmiston laadun kannalta erittäin keskeinen puute, joka liittyy ohjelman testaukseen. UiPath:ssa ohjelman testaaminen tapahtuu pääosin seuraamalla koko ohjelman läpimeno ja havaitsemalla poikkeamat. Tätä puutetta paikkaamaan pyrimme luomaan Microsoft Visual Studioon yksikkötestausympäristön, jossa yksittäisiä funktioita voidaan testata ja todeta toimiviksi. Tämä lisää huomattavasti ohjelmiston laatua, koska esimerkiksi Python-ohjelmakoodissa käsiteltävää XML-muotoista tietoa voidaan validoida ja todeta oikeaksi.

4.3 Prosessin soveltuvuuden arviointi

Ohjelmistorobotiikan hyödyntäminen Patriassa aloitettiin yleisellä arvioinnilla. Arvioinnissa tarkasteltiin, löytyykö organisaatiosta sellaisia prosesseja, jotka olisi mahdollista automatisoida ohjelmistorobotilla. Yleisessä prosessien arvioinnissa olivat mukana ne henkilöt, joilla on paras osaaminen yrityksen prosesseista. Tämän jälkeen yleisessä arvioinnissa tunnistettuja mahdollisesti automatisoitavia prosesseja tarkasteltiin yksityiskohtaisemmin kyseisen prosessin omistajan kanssa. Tässä vaiheessa kriittistä on ohjelmistoosaajan näkemys kyseistä prosessista, sillä hän kykenee arvioimaan prosessin automatisoinnin vaativuutta ja sitä onko automatisointi ylipäätään mahdollista toteuttaa. Mikäli tässä vaiheessa prosessi todetaan edelleen potentiaaliseksi kohteeksi toteuttaa ohjelmistorobotiikalla, on syytä ryhtyä pohtimaan automatisoinnilla saavutettavia hyötyjä liiketoiminnallisesta näkökulmasta. Automatisoinnilla saavutettavien hyötyjen tulee luonnollisesti olla sen aiheuttamia kustannuksia suuremmat. Kun prosessin automatisoinnin liiketoiminnallinen kannattavuus on arvioitu ja todettu, voidaan ohjelmistorobottia ryhtyä toteuttamaan.

Patriassa automatisoitavien prosessien suorituskykyä päätettiin mitata kolmella mittarilla, jotka ovat läpimenoaika, lopputuloksen laatu sekä säästynyt työaika. Nämä mittarit ovat vahvasti sidoksissa toisiinsa, eikä yksittäisen mittarin tarkastelu ole järkevää. Itse asiassa ainoastaan yhden mittarin tarkasteluun liittyy osaoptimoinnin vaara. Osaoptimointi tarkoittaa sitä, että jos prosessia kehitetään ainoastaan tarkastelemalla yhtä mittaria, saattaa muilla mittareilla mitattuna prosessin suorituskyvyssä tapahtua heikentymistä. Vaikka jonkin prosessin kohdalla todettaisiin, että automatisoimisella säästynyt työaika ei ole merkittävä, saattaa parannus esimerkiksi laadun osalta olla merkittävä, ohjelmistorobotin vähentäessä ihmillisen virheen todennäköisyyttä huomattavasti.

Aiemmin kirjallisuusosuudessa käytiin läpi potentiaalisten automatisoitavien prosessien arviointia. Oleelliset kysymykset, kun ohjelmistorobotiikkaa ryhdytään soveltamaan johonkin tiettyyn prosessiin, on esitetty seuraavassa taulukossa:

Taulukko 3. *Prosessin arviointi.*

Toistuuko prosessi usein? Viekö prosessin suorittaminen merkittävästi aikaa?
Onko prosessi työläs? Sisältääkö prosessi manuaalisesti tehtynä työläitä vaiheita?
Onko inhimillisen virheen todennäköisyys suuri? Ilmeneekö prosessissa ihmisen toiminnasta johtuvia virheitä, kuten kirjoitusvirheitä?
Mitkä ovat tietojärjestelmän, johon ohjelmistorobottia on tarkoitus soveltaa, nykytila ja tulevaisuuden näkymät? Onko tietojärjestelmään tulossa sellaisia muutoksia, jotka tekevät kyseisen manuaalisen työn jatkossa tarpeettomaksi?
Voidaanko prosessille määritellä yksiselitteiset säännöt? Vaikka koneoppivia järjestelmiä onkin jo luotu, puhuttaessa ohjelmistorobotiikasta ei tarkoiteta kognitiivisiin toimintoihin kykeneviä ohjelmistoja. Tämän lisäksi poikkeuksien määrän tulee olla rajallinen.
Toteutuuko prosessissa vaarallisen työn yhdistelmä? Kirjanpitoon liittyvissä prosesseissa on kyse yrityksen talouteen liittyvistä prosesseista. Vaarallisen työn yhdistelmä tarkoittaa sitä, että ohjelmistorobottia toteuttava henkilö saa mahdollisuuden sekä luoda maksu, että hyväksyä se. Tällaista tilannetta ei saa päästä syntymään ohjelmistorobotteja käytettäessä.

Mikäli vastaukset yllä esitettyihin kysymyksiin (Taulukko 3.) ovat myönteisiä, eikä tietojärjestelmää johon ohjelmistorobottia suunnitellaan olla radikaalisti muuttamassa tai korvaamassa, voidaan edellytykset ohjelmistorobotiikan soveltumiselle katsoa täyttyneeksi kyseisen prosessin osalta. Lisäksi tulee käydä läpi vaarallisen työyhdistelmän mahdollisuus ja todeta sen olevan mahdotonta.

Seuraavassa on esitelty (Taulukko 4.) esimerkki sellaisesta prosessista, joka todettiin kannattamattomaksi automatisoida, sillä hetkellä olevilla valmiuksilla. Prosessin suorittamisen kestoksi manuaalisin menetelmin arvioitiin noin 10 minuuttia. Kyseinen prosessi suoritetaan yhden kerran työpäivän aikana. Ihmisen suorittamana prosessissa esiintyvien virheiden todennäköisyys oli alhainen, koska kyseessä oli suurimmaksi osaksi kopiointi- ja tiedonsiirtotehtävä. Lisäksi oli tiedossa, ettei prosessia tulla suorittamaan jatkossa kovinkaan pitkään, korkeintaan muutaman kuukauden ajan. Siinä vaiheessa, kun kävimme prosessin kulkua läpi yhdessä asiakkaan kanssa, oli selvää, että sen automatisointi käytössä olevilla menetelmillä olisi mahdollista. Samalla myös todettiin, että se ei olisi kannattavaa, koska automatisointiin arvioitiin kuluvan liikaa resursseja suhteessa sillä saavutettaviin hyötyihin.

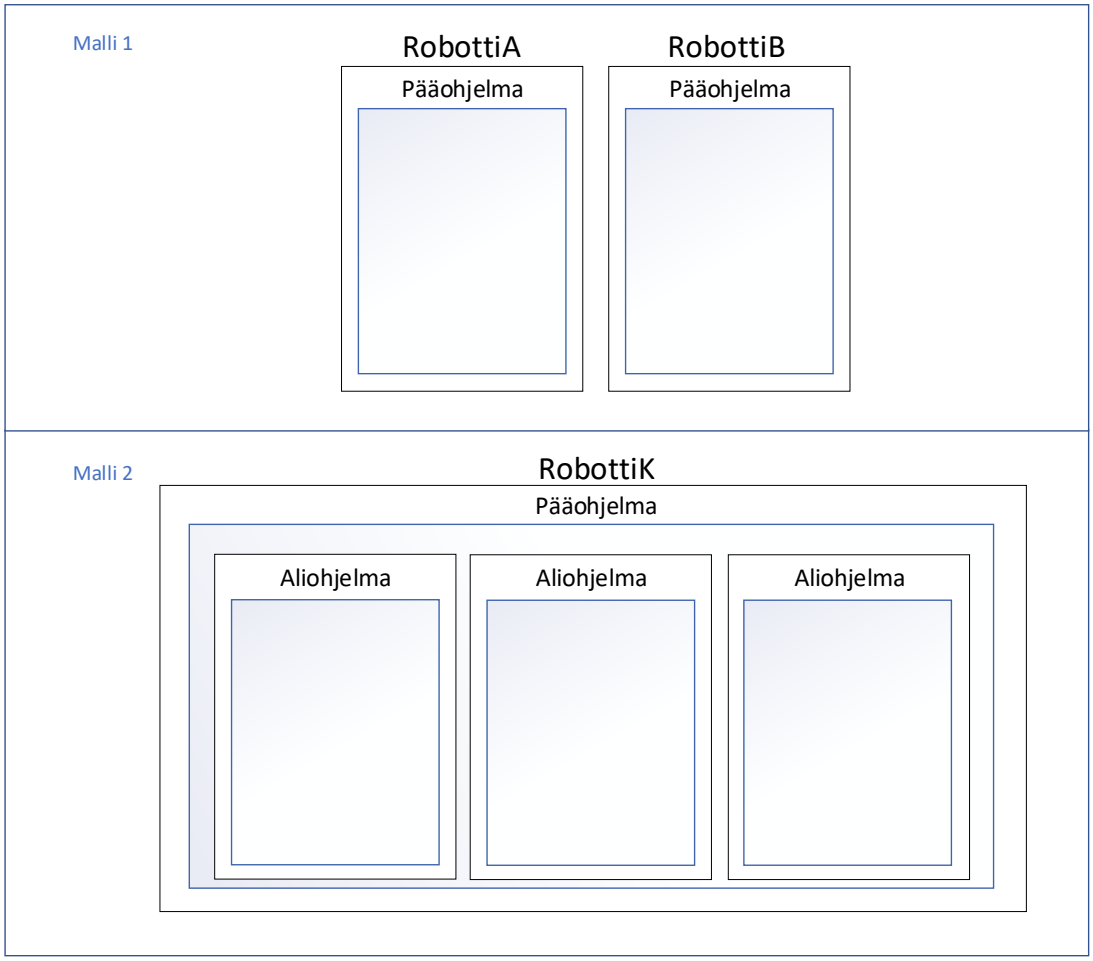
Taulukko 4. *Esimerkkitapaus prosessista, joka ei täyttänyt kriteerejä.*

	Esimerkkitapaus, jota ei automatisoitu
Toistuuko prosessi usein?	Kyllä, kerran päivässä
Onko prosessi työläs?	Ei ole
Onko inhimillisen virheen todennäköisyys suuri?	Ei, koska tietoja vain kopiotiin
Mitkä ovat tietojärjestelmän, johon ohjelmistorobottia on tarkoitus soveltaa, nykytila ja tulevaisuuden näkymät?	Ilmeni, että tilanne jatkuu tällaisena vain muutaman kuukauden
Voidaanko prosessille määritellä yksiselitteiset säännöt?	Kyllä, prosessille voitiin määritellä selkeät säännöt
Toteutuuko prosessissa vaarallisen työn yhdistelmä?	Ei toteutunut

4.4 Ohjelmistorobotin määrittely

Ohjelmistoarkkitehtuurin suunnittelun tiedetään olevan oleellinen ohjelmiston laatuun vaikuttava tekijä. Toinen tärkeä huomioon otettava tekijä on ohjelmakoodin muokattavuus ohjelmiston ylläpitovaiheessa, jotta ohjelmakoodiin muutoksia tehtäessä ei aiheuteta uusia virheitä ohjelman toiminnallisuuteen. Mikäli ohjelmistorobotiikan ohjelman hierarkiasta tehdään sen mukainen kuin sen toiminnallisuus on, muodostuu ohjelman rakenteesta selkeämmin ymmärrettävä kuin tilanteessa, jossa tätä näkökulmaa ei otettaisi huomioon [25].

Ohjelmistorobotiikkaprojektin alussa siihen liittyvien käsitteiden hahmottaminen oli hieman hankalaa. Tämä kuitenkin selkiytyi ajan mittaan. Hämmennystä on aiheuttanut ohjelmistorobotin käsite ja erityisesti se, että tarkoitetaanko sillä yhtä suoritettavaa ohjelmakoodia vai voiko yksi ohjelmistorobotti sisältää monia suoritettavia ohjelmakoodeja eli toimintoja suorittavia luokkia. Aiemman kokemuksen ja kirjallisuuden myötä todettiin, että riittävän samankaltaiset prosessit kannattaa laittaa yhden ohjelmistorobotin tehtäväksi, koska muuten robottien määrä kasvaa kohtuuttoman suureksi. Ohjelmistorobotin määrittely on oleellista, jotta yksittäinen ohjelmakoodi ja ohjelmistorobotti eivät aiheuta esimerkiksi keskusteltaessa väärinymmärryksiä.



Kuva 8. *Ohjelmistorobotin ohjelmarakenne.*

Ohjelmistorobottien ohjelmien rakenne noudattelee kahta päätyyppiä. Ne on esitelty yllä olevassa kuvassa (Kuva 8.). Mallissa yksi, yhdellä ohjelmistorobotilla tarkoitetaan yhtä ohjelmakoodia. Sen sijaan mallissa kaksi yksi ohjelmistorobotti sisältää pääohjelman, josta kutsutaan tehtäviä suorittavia aliohjelmia. Molemmille yllä esitellyille malleille löytyy tapauksia, joihin niitä kannattaa soveltaa. Esimerkiksi mallia yksi voidaan soveltaa sellaisten prosessien automatisointiin, joissa on loogista pitää ohjelman perusrakenne samana prosessista toiseen. Mikäli tällaisessa tapauksessa ohjelmarakenne toteutetaan mallin kaksi mukaan, muodostuu ohjelman ylläpidosta ja päivittämisestä työläs prosessi, koska silloin jokaiseen ohjelmaan joudutaan tekemään muutokset erikseen. Näin ollen järkevämpää on tehdä pääohjelmaan sellainen ohjelmakoodi, jolla on kykyä mukautua esimerkiksi eri yhtiöiden toiminnanohjausjärjestelmiin, niiden ollessa hyvin samankaltaisia. Mallia kaksi puolestaan voidaan soveltaa sellaisiin prosesseihin, joissa lähtöaineisto on sama, mutta prosessien vaiheet poikkeavat toisistaan merkittävästi. Ensimmäinen malli soveltuu lähinnä kaikista yksinkertaisimpien ohjelmistorobottien toteuttamiseen, jossa ei juurikaan esiinny poikkeuksia tai ehtoja.

4.5 Dokumentointi

Manuaalisten työtehtävien siirtyessä ihmisiltä tietokoneille häviää ihmisiltä myös osaaminen ajan kuluessa, koska kyseisiä prosesseja ei tarvitse enää tehdä manuaalisesti ja taito unohtuu. Mikäli ohjelmistoroboteissa esiintyy virhetilanteita, eikä ohjelman suoritus syystä tai toisesta onnistu, voi olla, että ihmisen tulee suorittaa se. Tästä syystä ohjelmistorobottien työnkulku tulee dokumentoida, sillä tarkkuudella, että tehtävä osataan suorittaa myös manuaalisesti ohjelmistorobotin ajautuessa virhetilanteeseen. Pidemmälle vietynä: jos yrityksestä ei prosessien automatisoinnin seurauksena enää löydy riittävää osaamista prosessien manuaaliseen läpivientiin, on tällä luonnollisesti negatiivisia vaikutuksia yrityksen operatiiviseen tehokkuuteen ja tätä kautta kustannuksiin. Käytettävä sovellusteknologia mahdollistaa itsessään melko hyvän prosessin dokumentoinnin, koska se sisältää visuaalisia elementtejä kuten kuvia. Lisäämällä riittävästi kommentteja ohjelmakoodista saadaan ymmärrettävä. Tämä vähentää, mutta ei poista tarvetta dokumentoida prosesseja esimerkiksi Microsoft Visio -työkalulla.

4.6 Käyttöönotto ja aikataulutus

Käyttöönoton suunnittelussa tulee ottaa huomioon ajankohta, jolloin ohjelmistorobotin on tarkoitus suorittaa ohjelmansa, koska ainoastaan yksi ohjelmistorobotti kerrallaan voi olla suorituksessa. Myöhemmin kun ohjelmistorobotteja on enemmän ja niiden luotettavuustaso paranee, eli virheiden suhteellinen lukumäärä onnistuneisiin suorituskertoihin saadaan järkevälle tasolle, hyödynnetään myös työajan ulkopuolista aikaa. Aikataulutus tulee tehdä niin, että ohjelmistorobotin asiakkaalla on sen tuottama lopputulos käytössä määritellysti. On oleellista määritellä asiakkaan kanssa, milloin ohjelmistorobotin tulee suorittaa kulloinkin kyseessä oleva ohjelma. Lisäksi on syytä huomata, että epäluotettavimmat ohjelmat ajetaan valvotusti ja luotettavammat ohjelmat voidaan suorittaa pienemmällä valvonnalla.

Ohjelmistorobotteja ajastetaan käyttämällä Windows Task Scheduleria. Kunkin ohjelmistorobotin suoritusaikataulu käydään läpi yhdessä kyseisen ohjelmistorobottia hyödyntävän asiakkaan kanssa. Koska ohjelmistorobotteja voidaan ajaa vain yksi kerrallaan, on kaikkien käytössä olevien ohjelmistorobottien suoritusaikataulua koordinoitava hallitusti. Patriassa tämä tapahtuu käyttämällä ohjelmistorobottia varten luotua Outlook-käyttäjätiliä. Outlook-tilin kalenteriin määritellään aikataulu, jolloin kunkin ohjelmistorobotin ohjelma suoritetaan. Tässä tarkastelussa on oleellista selvittää asiakkaiden kanssa, milloin ohjelmistorobotin on todellisuudessa tarve suorittaa tietty ohjelmakierto ja milloin sen tuottaman lopputuloksen on oltava asiakkaiden käytössä. Tätä on toisinaan syytä kyseenalaistaa, jotta saadaan asiakkaat pohtimaan, onko ohjelmistorobotin tuottaman lopputuloksen oltava käytössä aivan samalla tavoin kuin manuaalisen prosessin suorituksen. On syytä huomata, että ohjelmistorobottien ohjelmia voidaan ajaa myös yöaikaan, jolloin haluttu lopputulos on aamulla valmis ja asiakas saa tiedon töihin tultuaan.

Lisäksi on syytä huomata, että koordinoidessa ohjelmistorobottien aikataulusta tulee olla riittävän selkeä käsitys niiden ohjelmien vaatimista todellisista läpimenoajoista. On tyypillistä, että ohjelmistorobotin ohjelmien läpimenoajat vaihtelevat, riippuen useista eri tekijöistä kuten maksujen lukumäärästä tai käyttöliittymän viiveistä. Ohjelmistorobotiikkan soveltamista pidemmälle vietynä tässä olisi yksi mahdollinen jatkotutkimuskohde, jossa tilastoitaisiin käytössä olevien ohjelmistorobottien suoritusaikoja, jotta niille saadaan tarkempi arvio, Tässä kohtaa oleelliseksi seikaksi nousee aiemmin esitelty ohjelmistorobotin määrittäminen, koska jokainen ohjelmakierto vaatii jonkin verran ylimääräistä aikaa ennen kuin seuraava voidaan suorittaa, jotta varmistutaan siitä, että aiempi ohjelma on saatu suoritettua loppuun asti. Näin ollen, mikäli yksi ohjelmistorobotti määritellään liian suppeaksi kokonaisuudeksi johtaa tämä pitkällä tähtäimellä ohjelmistorobottiresurssin tehottomaan käyttöön sekä ohjelmistorobottien hallittavuuden heikkenemiseen niiden lukumäärän kasvaessa kohtuuttoman suureksi.

Ohjelmistorobottien asiakkaat määrittävät ajankohdat, jolloin ohjelmistorobottien tulee prosesseja suorittaa. Kuten jo aiemmin mainittiin, ohjelmistorobotteja ajastetaan käyttämällä Windows Task Scheduler-työkalua. Kyseiselle työkalulle on mahdollista syöttää parametrit XML-muotoisena tietona. Puhuttaessa parametreistä tarkoitetaan päivämäärää ja kellonaikaa, jolloin ohjelmistorobotti halutaan käynnistää. Tämän ansiosta aikataulutusta voisi jatkossa harkita ulkoistettavan ohjelmistorobotiikasta vastaavalta taholta asiakkaalle, näin erityisesti mikäli ohjelmistorobotin käynnistämisajankohta vaihtelee. Asiakas voisi tilata suorituksen vertaamalla ohjelmistorobotin Outlook-kalenteria ja omia tarpeitaan. Käynnistyksen toteuttava XML-tiedosto voitaisiin luoda jollain työkalulla, johon asiakas syöttää parametrit. Näin ollen ohjelmistorobotiikkaa ylläpitävä taho saisi tiedon käynnistyksestä ja voisi syöttää tiedot järjestelmään, jolla ohjelmistorobotteja ylläpidetään. Tässäkin oleellista on määrittää ohjelmistorobotin Outlook-kalenteriin muiden ohjelmistorobottien suoritusajat riittävällä tarkkuudella, jotta useita ohjelmistorobotteja ei vahingossa yritetä suorittaa samanaikaisesti.

5. TAPAUSTUTKIMUKSET

Ketterien menetelmien soveltuvuutta Patrian ohjelmistorobotiikan viitekehyksessä päätettiin kokeilla toteuttamalla tapaustutkimuksia, jotka esitellään seuraavaksi. Tapaustutkimuksia toteutettiin kaksi ja ne olivat keskenään hyvin erilaisia. Kuten kirjallisuusselvityksessä (Luku 2) todettiin, aloitetaan ohjelmistorobottien soveltaminen potentiaalin arvioinnilla. Tämän arvioinnin perusteella valitaan automatisoitava prosessi. Valmista viitekehystä tai toimintamallia ohjelmistorobotiikan soveltamiselle ei ollut, joten sitä ryhdyttiin rakentamaan. Prosessien automatisointi ohjelmistorobotiikalla aloitetaan pitämällä palaveri asiakkaiden kanssa, jotka tuntevat kyseisen prosessin kulun. Jotta ohjelmistorobotiikan soveltamisessa päästään sujuvasti alkuun, on käytävä läpi seuraavat seikat:

- Dokumentoidaan prosessin vaiheet sillä tarkkuudella, että dokumentin perusteella on mahdollista aloittaa ohjelmointi.
- Kartoitetaan tietojärjestelmiin sekä levyjaoille tarvittavat pääsyoikeudet ja tehdään niistä pyynnöt Patrian asiakastukeen.
- Pohditaan alustavasti mahdollisia testitapauksia, joilla ohjelmistorobottia myöhemmin testataan.
- Selvitetään asiakkaan kanssa ohjelmistorobotilta mahdollisesti vaadittavien tiedostojen sisältö ja niiden tallennuspaikka.

Aloituspalaverissa on tarkoitus saada yhteinen näkemys asiakkaan ja toimittajan välille automatisoitavasta prosessista ja sen vaatimuksista. Näitä vaatimuksia tarkennetaan projektin edetessä, mutta varsinkin ohjelmistorobottien kehityksestä vastaavan toimittajan kannalta oleellista on ymmärtää prosessi kokonaisuudessaan, jotta hän tunnistaa prosessin keskeisimmät elementit. Totesimme ohjelmistorobottien kehittämisen yhdeksi merkittävimmäksi pullonkaulaksi pääsyoikeuksien saamisen, joten mitä aikaisemmassa vaiheessa tiedetään mitä pääsyoikeuksia ohjelmistorobotin suorittamat prosessit vaativat, sitä sujuvammin projektissa päästään etenemään. Pääsyoikeuksien kartoittaminen saattaa aluksi olla hankalaa, koska ei tarkkaan tiedetä mihin kaikkiin tietojärjestelmän osiin ohjelmistorobotin tulee päästä käsittelemään tietoa. Lisäksi pääsyoikeuksien pyytäminen Patrian asiakastukeen ei ole kovin suoraviivainen toimenpide, varsinkin jos ei tiedä tarkalleen mitä pääsyoikeuksia pyytää.

5.1 Tapaus ROB0001 – Laskujen käsittely

Ensimmäisen ohjelmistorobotin suunnitteleminen alkoi kehitystyössä mukana olevien henkilöiden nimeämisellä, eli selvitettiin, kuka tuntee kyseessä olevan prosessin kaikkein

parhaiten. Tämän jälkeen kyseisten henkilöiden kanssa ryhdyttiin määrittelemään vaatimuksia, joita ohjelmistorobotin tulee kyetä toteuttamaan. Kyseessä on kahden toiminnanohjausjärjestelmän välinen tiedonsiirto, jossa kirjataan ja tarkastetaan erilaisia laskuja. Työssä on paljon manuaalisia vaiheita ja se toistuu usein, joten automatisoinnin potentiaalisten hyötyjen todettiin olevan merkittävät jo tämän perusteella.

Päätettäessä ohjelmistorobotin soveltamisesta kyseiseen prosessiin oleellisessa roolissa oli vertailu ohjelmistorobotilla ja varsinaisella järjestelmäintegraatiolla toteutetun ratkaisun välillä. Hyvin nopeasti selvisi, että järjestelmäintegraatiota tarjoavan toimittajan tarjous on niin kallis, eikä luottamus toimittajaan ole riittävän hyvällä tasolla, että järjestelmäintegraatiota edes harkittaisiin. Ohjelmistorobotiikalla tehtävää toteutusta päätettiin kokeilla. Ainakin omalta osaltani alusta asti vaikutti melko selkeältä, että toteutus olisi ylipäätään mahdollinen ohjelmistorobotiikalla tehtäväksi.

Taulukko 5. *Prosessin soveltuvuuden arviointi.*

	Tapaus ROB0001
Toistuuko prosessi usein?	Kyllä, toistuu päivittäin
Onko prosessi työläs?	Kyllä, prosessissa on paljon manuaalisia työvaiheita
Onko inhimillisen virheen todennäköisyys suuri?	Kyllä, koska tietoja syötetään manuaalisesti
Mitkä ovat tietojärjestelmän, johon ohjelmistorobottia on tarkoitus soveltaa, nykytila ja tulevaisuuden näkymät?	On epätodennäköistä, että tietojärjestelmää vaihdetaan lähitulevaisuudessa
Voidaanko prosessille määritellä yksiselitteiset säännöt?	Voidaan, mutta poikkeuksia on paljon
Toteutuuko prosessissa vaarallisen työn yhdistelmä?	Tällä hetkellä ei toteudu

Yllä olevaan taulukkoon (Taulukko 5.) on koottu prosessin soveltuvuuden arvioinnin tulokset. Kysymyksissä yhdistyy kaksi näkökulmaa, joista ensimmäinen on kirjallisuusselvityksessä esitellyt automatisoitavan prosessin kriteerit ja toinen on Patrian viitekehyksessä esiintyvät oleelliset kysymykset. Näiden lähtötietojen perusteella kyseessä olevaa laskujen käsittelyprosessia päätettiin ryhtyä toteuttamaan ohjelmistorobotiikalla.

Ohjelmistorobottien soveltamisessa yksi keskeisimpiä näkökulmia on sen tuoma liiketoiminnallinen hyöty. Alla olevasta taulukosta (Taulukko 6.) nähdään käsiteltävien laskujen lukumääriä vuosina 2017 ja 2018 tähän asti. PO- ja RO -lyhenteet tarkoittavat ostotilauksellisia laskuja, KULU -lyhenne tarkoittaa kululaskuja ja HYV -lyhenne puolestaan merkitsee hyvityslaskuja, jotka rajattiin alustavasti tämän automatisoinnin ulkopuolelle. Taulukosta (Taulukko 6.) nähdään, että suurin volyymi on kululaskuilla, joten laskujen käsittelyn automatisointi päätettiin aloittaa kyseisestä laskutyypistä. Yhden kululaskun käsit-

telyn todettiin vievän keskimäärin neljä minuuttia manuaalisesti käsiteltynä. Ostotilauksellisia laskuja voidaan käsitellä huomattavasti nopeammin, koska niitä ei kirjata tietojärjestelmään, niiden tiedot vain tarkastetaan. Tässä työssä ei tarkastella liiketoiminnallisia hyötyjä sen syvällisemmin, mutta on oleellista ymmärtää, että laskujen lukumäärien ollessa tuhansia ja käsittelyajan minuutteja kuluu prosessiin huomattavia määriä työaikaa.

Taulukko 6. *Tilastoa käsiteltävien laskujen lukumääristä.*

Laskujen lukumäärät				
Laskutyyppi	2017		2018	
PO	2047	41 %	1525	46 %
RO	185	4 %	126	4 %
KULU	**2639**	**53 %**	**1594**	**48 %**
HYV	68	1 %	71	2 %
Yhteensä	4939		3316	

Aluksi selvitettiin vain prosessin keskeisimmät vaiheet, jotta varsinainen ohjelmointivaihe voitiin aloittaa mahdollisimman pian. Ohjelmistonkehityksen edetessä vaatimuksia ilmeni luonnollisesti lisää, koska asiakkaat eivät lähtötilanteessa muistaneet määritellä kaikkea, eikä kaikkea dataakaan ollut heti saatavilla. Tämä ei kuitenkaan ollut ongelma, koska prosessin keskeisimmät piirteet käytiin läpi tarkasti ja yksityiskohtien lisääminen tähän kokonaisuuteen vaikutti hyvältä tavalta kehittää ohjelmistoa. Mielestäni projektin alussa onkin tärkeää määritellä automatisoitava prosessi riittävällä tarkkuudella sekä tunnistaa sen keskeisimmät vaiheet, joiden ympärille voidaan rakentaa myöhemmin lisää toiminnallisuuksia. Varsinainen ohjelmointi kannattaa aloittaa aikaisessa vaiheessa, eikä vasta hyvin yksityiskohtaisen vaatimusten määrityksen jälkeen. Ohjelmoidessa prosessia sen ymmärtää paremmin, jonka ansiosta asiakkailta osaa kysyä oleelliset tarkentavat kysymykset.

Käsiteltävät laskut ovat XML-muotoisia. Oli luonnollista aloittaa ohjelmiston kehittäminen tarkastelemalla XML-muodossa olevan laskun rakennetta. Kaikkien kenttien arvoja ei ollut tarvetta käyttää, vain edellisessä vaiheessa määriteltyjen vaatimusten mukaan XML-tiedostosta haetaan kohteena olevaan toiminnanohjausjärjestelmään syötettävät tiedot tietyistä kentistä. Ohjelman rakenteen selkeydyn kannalta ja osittain kokeilumielessä päätin ohjelmoida ohjelmistorobotille sisään luettavan XML-muotoisen laskun tietojen hakemisen Python-ohjelmointikielellä. UiPath-sovellusteknologiaa käytetään tässä tapauksessa ainoastaan käyttöliittymän rajapinnan läpi ohjelmointiin ja sen käyttämää tietoa käsitellään UiPath-sovelluksen näkökulmasta ulkoisesti. UiPath:lla toteutettu ohjelma kutsuu näitä tietoa käsitteleviä skriptejä tietyissä kohdissa ohjelmakiertoa.

Tehdessäni tätä ohjelmistorobottia oivalsin, että tässä olisi Patrialla hyvä tilaisuus mahdollisesti pilotoida koneoppivaa järjestelmää tulevaisuudessa. Tulevaisuudessa siksi, että laskuista kerättyä sopivaa dataa ei ole vielä saatavilla, mutta sitä voitaisiin kerätä tämän ohjelmistorobotin avulla. Ohjelmistorobotti hoitaisi luokittelun samalla kun se kirjaa laskuja järjestelmään. Idea syntyi kyseessä olevan toiminnanohjausjärjestelmän epäluotettavasta käyttäytymisestä. Virheitä esiintyy satunnaisissa kohdissa tuntemattomista syistä. Datan käsittely voisi antaa näiden juuri syistä jotain viitteitä.

Ajatuksena on kerätä laskuista dataa ja luokitella se sellaisiin laskuihin, joiden kirjaaminen onnistui ja sellaisiin, joiden kirjaaminen ei onnistunut. Kun dataa on saatu talteen riittävän pitkältä ajalta, voidaan luoda malli joka ”koulutetaan” kerätyllä datalla. Kun malli on ”koulutettu”, sitä voitaisiin käyttää ennustamaan, onnistuuko entuudestaan tuntemattoman laskun käsittely ohjelmistorobotilta. Tätä voitaisiin hyödyntää validoimaan laskuja ennen kuin niitä syötetään ohjelmistorobotille. Käytännössä malli palauttaisi todennäköisyyden sille, että lasku menee läpi. Mikäli todennäköisyys on riittävän suuri, sallitaan kyseisen laskun ajaminen ohjelmistorobotilla.

Tällaisen mallin käyttämien lisäisi todennäköisesti ohjelmistorobotin luotettavuutta, mikäli virheelliset laskut voitaisiin suodattaa pois ohjelmistorobotin lähtöaineistosta. Tämä toimisi myös käytännön sovelluksena koneoppivasta mallista ja sen myötä saatua tietämystä voitaisiin käyttää mahdollisesti myös muissa sovelluskohteissa. Vaikka mallia ei koskaan toteutettaisiin tai otettaisi käyttöön, mahdollisuus hyödyntää sellaista ratkaistaan ohjelmistorobotin kehitysvaiheessa, koska ohjelmistorobotti voidaan määritellä tekemään datan luokittelun valmiiksi, joka on yksi datan hyödyntämisen työläimpiä prosesseja. Toisin sanoen ohjelmistorobotin kehitysvaiheessa tehdään määrittely sille, että ohjelmistorobotti osaa tallentaa laskun datan perään merkin, joka indikoi laskun menneen joko prosessin läpi tai jääneen virheeseen.

Tapaus ROB0001 osoittautui paljon haastavammaksi kuin aloituspalaverin perusteella ymmärrettiin. Kyseisen ohjelmistorobotin on tarkoitus hoitaa ainoastaan pieni osa kahden toiminnanohjausjärjestelmän välisestä integraatiosta. Ohjelmistorobotin rooli tässä kokonaisuudessa on varsin pieni, mutta silti kriittinen. Mikäli prosessin laajuus olisi ollut selvillä heti projektia aloitettaessa olisi esimerkiksi ohjelmistorobotin sovellusarkkitehtuuri osattu määritellä eri tavoin kuin mitä nyt tehtiin. Vaatimuksia tuli jatkuvasti lisää, eikä alussa luomani sovellusarkkitehtuuri enää jossain vaiheessa soveltunutkaan ratkaisuksi. Totesin suureksi riskiksi sen, että ohjelmistosta muodostuu liian kömpelö ylläpidettäväksi ja muokattavaksi vaatimusten määrän kasvaessa, joten muutin sovellusarkkitehtuuria täysin ainakin kolme kertaa kehitystyön aikana.

Käyttöönotto tapahtuu kokeilemalla ohjelmistorobottia muutamalla laskulla kerrallaan ja tarkastelemalla tulokset. Tähän mennessä testejä on suoritettu ainoastaan kyseessä olevan toiminnanohjausjärjestelmän testitietokantaan, joten on vielä toistaiseksi epäselvää, toimiiko tuotantotietokanta samalla tavalla ja tämä tulee ottaa huomioon käyttöönotossa.

5.2 Tapaus ROB0002 – Maksuehdotelmat ja maksulistat

Seuraavassa tapaustutkimuksessa toteutetaan Patrian kirjanpitoon liittyvän prosessin automatisointi. Kirjanpitoon liittyvät maksuehdotelmat, maksulistat sekä niiden pohjalta luotavat pankkiin lähetettävät tiedot suoritetaan kaksi kertaa viikossa, tiistaisin ja torstaisin. Kyseisen prosessin todettiin olevan manuaalisesti tehtynä työläs ja sen suorittamiseen kuluu aikaa neljä tuntia viikossa.

Taulukko 7. *Prosessin soveltuvuuden arviointi.*

	Tapaus ROB0002
Toistuuko prosessi usein?	Kyllä, kaksi kertaa viikossa
Onko prosessi työläs?	Kyllä, prosessissa on paljon manuaalisesti tehtynä hidasta
Onko inhimillisen virheen todennäköisyys suuri?	Kyllä, koska tietoja syötetään manuaalisesti
Mitkä ovat tietojärjestelmän, johon ohjelmistorobottia on tarkoitus soveltaa, nykytila ja tulevaisuuden näkymät?	On epätodennäköistä, että tietojärjestelmää vaihdetaan lähitulevaisuudessa
Voidaanko prosessille määritellä yksiselitteiset säännöt?	Voidaan määritellä ja poikkeuksien määrä on rajallinen
Toteutuuko prosessissa vaarallisen työn yhdistelmä?	Tällä hetkellä ei toteudu, mutta tähän tulee jatkossa kiinnittää huomiota, sillä pääsyoikeuksia vaadittiin runsaasti

Kuten edellinenkin ohjelmistorobotiikan sovelluskohde, tämäkin arvioitiin yllä olevassa taulukossa (Taulukko 7.) esitettyjen kysymysten perusteella. Kun nämä kohdat oltiin käyty läpi, vastauksia tarkastelemalla todettiin kyseinen prosessi soveltuvaksi ohjelmistorobotilla automatisoitavaksi.

Prosessissa kirjaudutaan Patrian erääseen toiminnanohjausjärjestelmään ja sieltä tallennettaan tilitiedot, jokaisen yrityksen osalta yhteiselle levyjaolle, PDF-muodossa. Tämä tapahtuu kuuden yhtiön osalta ja prosessi toistuu samanlaisena jokaisen yhtiön kohdalla. Yhtiöillä on useita tilejä eri valuutoille. Tiistaina kyseinen prosessi suoritetaan kaikille yhtiöille ja niiden kaikille eri valuuttatileille. Torstaina suoritetaan ainoastaan euromääräisten tilien maksujen käsittely.

Ohjelmistorobotiikan hyödyntäminen on Patriassa vielä alkuvaiheessa. On tärkeää arvioida, millaisia toiminnallisuuksia tullaan mahdollisesti tarvitsemaan tulevissa projekteissa, jotta aiemmin luotuja ohjelmakoodeja voidaan hyödyntää tehokkaasti. Tässä tapaustutkimuksessa ohjelmistorobotin oli määrä tuottaa yhteenvetotaulukko tallennettujen PDF-muotoisten tilisaldotietojen pohjalta. Totesimme, että tällainen toiminnallisuus tulee

todennäköisesti toistumaan myös myöhemmin toteutettavissa projekteissa, joten toiminnallisuus päätettiin toteuttaa Python-ohjelmointikielellä. Tässä ohjelmistorobotissa käyttöliittymän läpi tapahtuva ohjelmointi on toteutettu UiPath-sovellusteknologialla. Ohjelman lopussa kutsutaan yhteenvetotaulukon toteuttavaa Python-koodia, joka luo uuden Excel-tiedoston samaan polkuun, jonne UiPath-sovellus tallensi kyseisenä päivänä luodut maksulistat. Toiminnallisuudet on jaettu niin, että tiedon käsittely, jota ei ole tarkoitus toteuttaa käyttöliittymän rajapinnan kautta toteutetaan muilla työkaluilla, kuten Pythonilla tai C# ohjelmakoodeilla.

UiPath-sovellusteknologialla luodaan ohjelmistorobotin ohjelman runkorakenne, josta voidaan tarvittaessa kutsua muilla ohjelmointikielillä luotuja toiminnallisia komponentteja. Tämä lähestymistapa selkeyttää ohjelmistorobotin hierarkiaa, koska kuten aiemminkin todettiin, UiPath-sovellusteknologialla toteutettuna ohjelman rakenne muuttuu nopeasti hankalasti luettavaksi. UiPathin pääkäyttötarkoitus onkin rajattu pääsääntöisesti osuuksiin, joissa automatisointia toteutetaan käyttöliittymän rajanpinnan läpi, koska tämä on kyseisen sovellusteknologian selkeä vahvuus.

Tässä tapaustutkimuksessa toteutetun ohjelmistorobotin (ROB0002) käyttöönotto tehtiin vaiheittain. Käyttöönottovaiheessa yhteistyö asiakkaan kanssa korostuu entisestään, jotta ohjelmistorobotin toiminnasta saadaan välittömästi palautetta. Käyttöönotto tehtiin vaiheittain, aloittaen yksinkertaisemmasta tapauksesta. Ensimmäisessä käyttöönotetussa versiossa ohjelmistorobotti suoritti torstaisin tehtävän prosessin, joka on yksinkertaisempi kuin tiistaina suoritettava prosessi. Kun tämä saatiin toimimaan, laajennettiin seuraavaa versiota niin, että ohjelmistorobotti kykeni toteuttamaan myös tiistaina suoritettavan laajemman prosessin automaattisesti. Tarkoituksena oli saada nopeasti jokin toimiva versio asiakkaiden käyttöön, jotta he saavat paremman käsityksen ohjelmistorobotiikasta. Tämän jälkeen jatkettiin yhteenvetotaulukon generoivan ohjelmakoodin kehittämistä.

5.3 Tapaustutkimusten vertailu ja yhteenveto

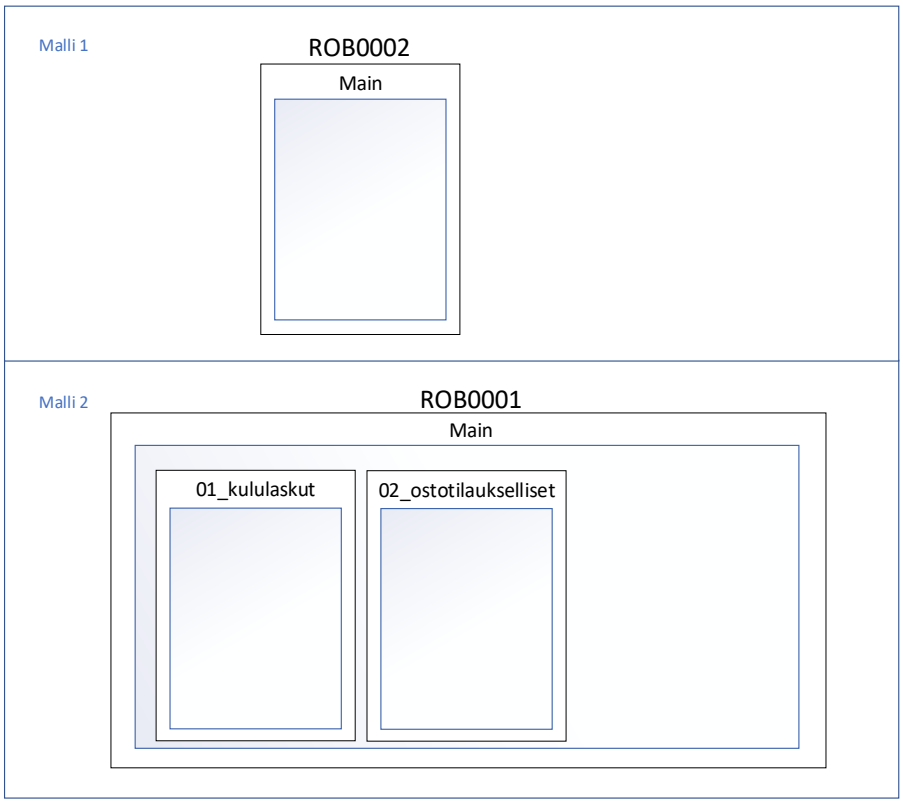
Kaikissa tapaustutkimuksissa ohjelmistorobotiikkaprojektin etenemisen yhdeksi suurimmaksi pullonkaulaksi muodostui projektiin osallistuvien henkilöiden, lähinnä asiakkaiden tavoittaminen ja tiedon saaminen heiltä. Toinen merkittävä pullonkaula ohjelmistorobottien soveltamisessa Patrian verkossa on tarvittavien pääsyoikeuksien kartoittaminen ja niiden pyytäminen. Pääsyoikeuksia johonkin tietojärjestelmään tai sen osaan pyydetään lähettämällä sähköpostilla pyyntö Patrian käyttäjätukeen, josta pyyntö ohjautuu siitä päättävälle taholle.

Taulukko 8. *Tapaustutkimusten yhteenveto.*

	Tapaus ROB0001	Tapaus ROB0002
Vaativuus	Erittäin vaativa	Kohtalainen
Hyödyt	Laskujen käsittelyn läpimenoajan arvioitiin puolittuneen	Työaikaa säästyy neljä tuntia viikossa
Riskit	Toiminnan ohjausjärjestelmä käyttäytyy epästabiilisti	Kuvan tunnistukseen perustuva ohjelmointi
Suorituskyky	Ohjelmistorobotti kykenee kirjaamaan ja tarkistamaan laskuja ihmistä nopeammin ja samalla inhimillisen virheen todennäköisyys pienenee	Ohjelmistorobotti suorittaa tehtävän nopeammin ja osaa generoida yhteenvedon
Huomioita	Ensimmäiseksi projektiksi tämä oli hyvin haastava ja laaja	Tähän projektiin oli paljon ohjelmallista toiminnallisuutta jo valmiina, joten toteutus oli suoraviivaista

Yllä olevaan taulukkoon (Taulukko 8.) on koottu yhteenveto toteutetuista tapaustutkimuksista. Tapaukset olivat keskenään hyvin erilaisia ja vaativat erilaisia lähestymistapoja niin ohjelman rakenteen kuin taustalla olevien tietojärjestelmien osalta.

Tapaus ROB0001 osoittautui erittäin vaativaksi ohjelmistorobotiikan sovelluskohteeksi, koska tietojärjestelmä, johon ohjelmistorobottia sovelletaan, käyttäytyy satunnaisesti. Tietojärjestelmä on rakenteeltaan Hypertext Markup Language (HTML) -pohjainen, jonka ansiosta sivun tunnistetiedot ovat ohjelmistorobotin näkyvissä. Näissä tunnistetiedoissa kuitenkin esiintyi satunnaisuutta ja ne saattoivat vaihtaa arvojaan suorituskerrasta toiseen. Tämä johtaa siihen, ettei ohjelmistorobotti tunnista sille ohjelmoitua reittiä siirtyä järjestelmässä esimerkiksi näkymästä toiseen. Edellä mainitun järjestelmän ominaisuuden löytyminen vei runsaasti testaamiseen. Lisäksi kyseisessä tietojärjestelmässä esiintyy virheilmoituksia, joiden alkuperä ei ole tullut selville. Nämä ominaisuudet olivat järjestelmän asiantuntijoille selvillä jo projektin alussa, joten osasimme odottaa haasteita ohjelmistorobotin ohjelmoimisessa. Tapaus ROB0001:n osalta tämän diplomityön tekemisen aikana ei ehditty käyttöönottovaiheeseen asti, koska meneillään oleva taustatyö eli järjestelmäintegraatio ei ehtinyt valmistua.



Kuva 9. *Tapaustutkimuksissa sovelletut ohjelmarakenteet.*

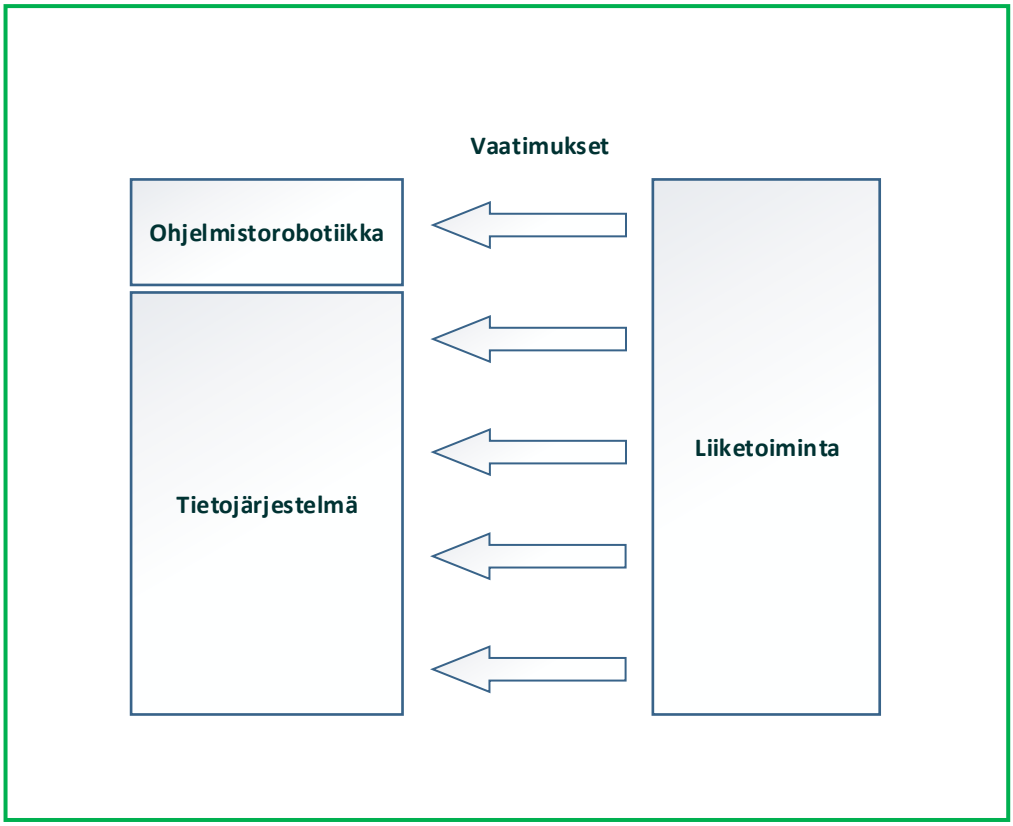
Tapaus ROB0002 ohjelmoitiin käyttämällä mallin yksi (Kuva 9.) mukaista ohjelmarakennetta. Ohjelma toistuu samankaltaisena kaikkien kuuden eri yhtiön toiminnanohjausjärjestelmien osalta, joten ohjelman jakaminen mallin kaksi (Kuva 9.) mukaisiin aliohjelmiin ei osoittautunut järkeväksi tavaksi jäsentää ohjelmaa. Tapauksen ROB0002 ohjelmistorobotin ohjelma perustuu kuvien mukaan ohjelmointiin, koska se on ainut keino tehdä ohjelmistorobotti kyseiseen tietojärjestelmään. Kuvien tunnistus tapahtuu näytön pikseleiden perusteella. Tämä luo haasteita, koska tunnistuksen luotettavuuteen vaikuttaa näytön ominaisuudet. UiPath-sovellusteknologialla on mahdollista määrittää tietty rajaarvo, kuinka tarkasti kuvan tulee näytöllä esiintyä, mutta tämän tarkkuus ei ole vielä täysin tiedossa.

Tapaustutkimusten perusteella ohjelmistorobottien toiminnallisuuksia kannattaa jakaa pienempii kokonaisuuksiin. Tämä korostuu, mikäli kyseessä oleva ohjelmistorobotti on rakenteeltaan monimutkainen. Varsinkin ohjelmistorobotin ROB0001 toiminnallisuuksien jakaminen oli kriittistä, koska jatkuvasti uusia vaatimuksia toteuttavien ominaisuuksien lisääminen on hankalaa, jos sovelluksen hierarkia ei ole selkeä. Toisaalta mikäli ohjelmistorobotin rakenne on yksinkertainen, sitä ei kannata välttämättä jakaa pieniin osiin. Kriittistä on hahmottaa sovellusarkkitehtuurin yhteensopivuus prosessin toiminnallisuuden kanssa. Tapaustutkimuksessa ROB0002 toteutetun ohjelmistorobotin käyttöönoton jälkeen ilmeni ohjelmistorobotin toiminnallisuudessa merkittäviä virheitä. Ohjelmistorobotin virheelliseen toimintaan löydettiin kaksi pääsyytä. Ensimmäinen oli kyseisen ohjelmistorobotin kehittäminen ennen kuin kaikki vaadittavat pääsyoikeudet oli myönnetty.

Tämä johti tilanteeseen, jossa tietojärjestelmän antamien virheilmoitusten alkuperä jäi epäselväksi ja ne tulkittiin tietojärjestelmän normaaliksi toiminnaksi. Toinen syy virheelliselle toiminnalle oli puutteet vaatimusten määrittelyssä. Tästä opimme, että ohjelmistorobotin kehittämistä ei tule aloittaa ennen kuin pääsyoikeudet on myönnetty ja vaatimukset on määritelty.

6. POHDINTA

Tässä luvussa käydään läpi tutkimustulokset ja muita ohjelmistorobotiikan osalta oleellisia näkökulmia, joita tutkimuksen myötä ilmeni. Tämän työn kirjallisuusselvityksessä mainittiin, että tällä hetkellä ohjelmistorobotiikka nähdään väliaikaisena ratkaisuna siirryttäessä vanhoista runsaasti manuaalista työtä vaativista tietojärjestelmistä uusiin aiempaa automaattisemmin toimiviin tietojärjestelmiin. Mielestäni tämä on yksi erittäin keskeinen näkökulma, joka tulee ottaa huomioon ohjelmistorobotiikkaa sovellettaessa, sillä jos sitä ei huomioida saattaa itse tietojärjestelmien sopivuuden jatkuva uudelleen arviointi jäädä yritykseltä tekemättä. Ohjelmistorobotiikkaa soveltamalla saadaan palaute, onnistuiko tietojärjestelmän vaatimusten määrittely. Kun tämä tieto otetaan huomioon suunniteltaessa uusia tietojärjestelmiä liiketoiminnallisiin tarpeisiin, saadaan vaatimusten määrittelyyn mukaan sellaista tietoa, jota ei muuten ehkä olisi osattu ottaa huomioon. Keskeiseksi tekijäksi nouseekin yrityksen eri osastojen välinen vuorovaikutus uusia tietojärjestelmiä suunniteltaessa, jotta järjestelmät vastaisivat mahdollisimman hyvin niille asetettuihin vaatimuksiin.



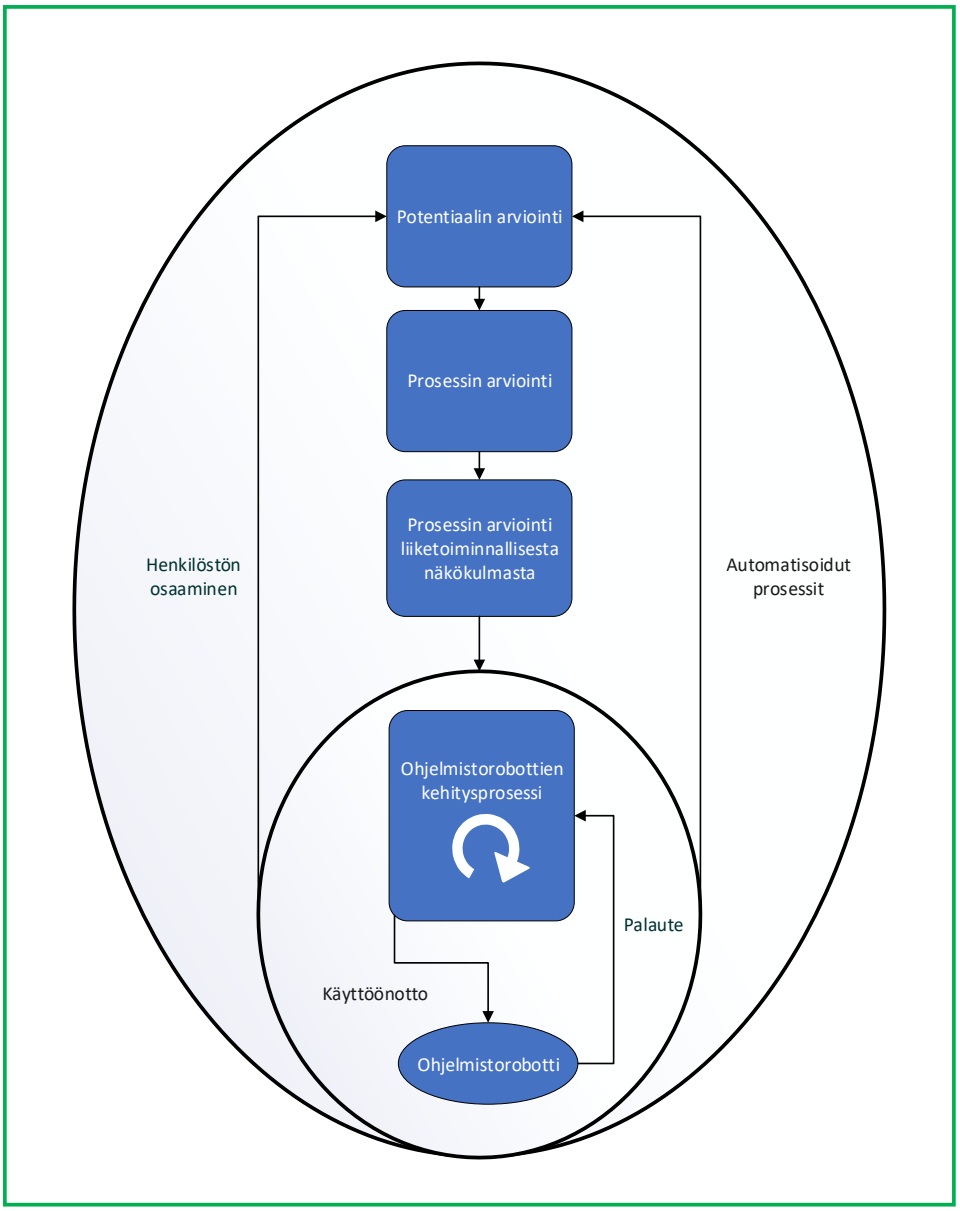
Kuva 10. *Ohjelmistorobotiikan sijoittuminen suhteessa tietojärjestelmiin.*

Yrityksen tietojärjestelmät palvelevat aina jotain tarvetta. Tarve on usein liiketoiminnallinen tai siihen kytköksissä. Tietojärjestelmien tulisi tukea yrityksen liiketoiminnallisia prosesseja mahdollisimman hyvin. Tietojärjestelmät, kuten toiminnanohjausjärjestelmät, saattavat olla monimutkaisia ja sisältää paljon muuttuvaa tietoa sekä toiminnallisuuksia. Tällaisen tietojärjestelmän vaatimusten määrittely saattaa olla hankalaa johtuen sen laajuudesta. Luvussa kolme mainittiin, että ohjelmistokehitysprosessin yksi hankalimpia vaiheita onkin vaatimusten määrittely. Alla olevassa kuvassa (Kuva 10.) on esitelty yrityksen liiketoiminnan suunnalta sen käyttämiin tietojärjestelmiin kohdistuvia vaatimuksia ja ohjelmistorobotiikan sijoittuminen tähän viitekehykseen. Huomataan, liiketoiminnan vaatimusten saattavat mennä osittain ohi tietojärjestelmän kyvystä vastata niihin. On loogista, että tämä luo kysyntää ohjelmistorobotiikalle.

6.1 Tutkimustulokset

Tutkimuksen alussa esitettiin tutkimuskysymys: Miten ohjelmistorobotteja kehitetään ja käyttöönotetaan ketterillä menetelmillä? Tähän kysymykseen etsittiin vastauksia toteuttamalla tapaustutkimuksia, joiden toteuttamisessa sovellettiin ketteriä ohjelmistokehitys menetelmiä. Seuraavalla sivulla esiteltävän viitekehyksen (Kuva 11.) on tarkoitus kytkeä yhteen niin ketterät menetelmät ohjelmistokehityksessä kuin automatisoinnin potentiaalin arviointi Patriassa. Kuten kuvasta (Kuva 11.) käy ilmi nämä prosessit muodostavat sisäkkäisiä syklejä, jonka keskiössä on ketteriin menetelmiin perustuva ohjelmistokehitysprosessi, jonka todettiin olevan luonteeltaan niin iteratiivinen kuin inkrementaalinenkin. Eli kehityksen eri vaiheita toteutetaan keskenään samanaikaisesti ja toiminnallisuuksia lisätään iteraatiokierroksilla. Viitekehyksessä (Kuva 11.) nivoutuvat yhteen tämän työn alussa esitelty näkemys tietojärjestelmien tutkimuksesta, ketterien menetelmien soveltaminen sekä ohjelmistorobotiikka. On ilmeistä, että ohjelmistorobotiikan vaikutukset eivät rajaudu ainoastaan itse ohjelmistorobotteihin vaan ne näkyvät organisaatiossa huomattavasti laajemmin, koskien niin henkilöstöä kuin yrityksen prosessejakin.

Kuten aiemmin todettiin, ohjelmistorobotiikan soveltamisen ensimmäinen vaihe on potentiaalisten sovelluskohteiden arviointi. Tätä seuraa tarkempi arviointi niin itse prosessin näkökulmasta kuin liiketoiminnallisesta näkökulmastakin. Kun prosessin arvioinnin vaiheet on käyty läpi ja todettu jokin prosessi mahdolliseksi ja kannattavaksi automatisoida. Siirrytään seuraavaan vaiheeseen, jossa ohjelmistorobotti kehitetään vaatimusten mukaan. Kuvasta (Kuva 11.) nähdään, että tätä seuraa kyseisen ohjelmistorobotin käyttöönotto, kun ohjelmistorobotin on todettu vastaavan sille asetettuihin vaatimuksiin. Ohjelmistorobotin kehittäminen muodostaa itsessään prosessin, jonka jossain vaiheessa ohjelmistorobotti otetaan käyttöön. Tästä saadaan palaute itse ohjelmistorobotin kehitysprosessille, jonka perusteella ohjelmistorobotin kehitysprosessia voidaan kehittää. Kun tätä jatketaan ja ohjelmistorobotiikalla toteutettujen sovellusten lukumäärä kasvaa myös organisaation tietämys aiheesta.



Kuva 11. *Patrian kyky tehdä ohjelmistorobotteja.*

Ohjelmistorobottien soveltamisesta saatavalla palautteella on kaksi ilmeistä ulottuvuutta (Kuva 11.). Henkilöstön osaaminen ja aiemmin automatisoidut prosessit. Kuten jo aiemmin todettiin, on ohjelmistorobottien toteuttaminen mitä suurimmassa määrin vuorovaikutusta ja yhteistyötä eri osapuolten välillä. Yhteistyöhön osallistuvien henkilöiden osaaminen ja tietämys ohjelmistoroboteista kasvaa luonnollisesti projektien toteuttamisen myötä. Mikäli tätä kertynyttä tietämystä sovelletaan uudestaan potentiaalin arvioinnissa, voidaan yrityksen prosesseja arvioida automatisoinnin kannalta aiempaa perusteellisemmin ja löytää sellaisia automatisoitavia prosesseja, joita ei aiemmalla arviointi kerralla välttämättä ymmärretty. Mikäli tätä näkemystä laajennetaan edelleen esimerkiksi uusien teknologioiden tarkastelulla, voidaan automaation kannalta potentiaalisten prosessien tarkastelussa päätyä toteuttamaan automatisointia entistä edistyneemmillä ratkaisuilla, ku-

ten koneoppimiseen perustuvilla sovelluksilla. Teknologinen kehitys luo uusia mahdollisuuksia yhdessä tietämyksen kanssa automatisoida aiempaa monimutkaisempia ja vaativampia prosesseja.

Patrian näkökulmasta ketteriin menetelmiin perustuvassa ohjelmistokehityksessä selkeä etu on menetelmien joustavuus, koska yrityksen sisäiset asiakkaat eli kirjanpitäjät ja muut ohjelmointiin perehtymättömät henkilöt eivät välttämättä osaa määritellä prosessiaan riittävällä tarkkuudella lähtötilanteessa, mistä aiheutuu jatkuvia muutoksia ohjelmistolle määriteltäviin vaatimuksiin. Lisäksi järjestelmät eivät ole niin monimutkaisia, että täsmällisen tarkkaa vaatimusten määrittelyä ohjelmistokehitysprojektin alussa kannattaisi edes vaatia. Ohjelmistokehitys on siis iteratiivista eli ohjelmistoa kehitetään jatkuvasti paremmin asiakkaan tarpeet täyttäväksi lisäämällä toiminnallisuuksia. Tärkeää on käydä vuoropuhelua yhdessä asiakkaiden kanssa ja selvittää automatisoitavan prosessin kulku aluksi pääpiirteissään, koska tämän jälkeen ohjelmointi voidaan aloittaa.

Kyseessä on inkrementaalinen ohjelmistokehitys, sillä enemmän kuin yhtä vaihetta voidaan tehdä yhtä aikaa. Inkrementaalinen ohjelmistokehitys on yksi ketterien menetelmien kulmakivistä. Aloituspalaverin jälkeen ohjelmistokehittäjä voi aloittaa ohjelmoinnin, kun samaan aikaan asiakas mahdollisesti määrittelee lisää vaatimuksia tai selvittää muita ohjelmistorobotilta vaadittavia toiminnallisuuksia. Kuten kirjallisuusselvityksessä todettiin, inkrementaalisen ohjelmistokehityksen etuna on useamman kuin yhden vaiheen tekeminen samaan aikaan, mikä säästää aikaa. Ketteriin menetelmiin perustuvan ohjelmistokehityksen 12 keskeisintä periaatetta esiteltiin luvussa kolme. Seuraavassa kappaleessa on nostettu esille tapaustutkimusten perusteella esiinnousseita seikkoja kyseisten menetelmien soveltamisesta Patrian viitekehyksessä.

“Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.”

Jokaisessa tehdyssä tapaustutkimuksessa ohjelmistorobotille kohdistuneet vaatimukset muuttuivat projektien edetessä, joten tämä oli ehkä keskeisin seikka, jonka perusteella ketterät menetelmät valittiin Patriassa ohjelmistokehitysmenetelmäksi. Lisäksi on syytä muistaa, että Patrian ympäristössä ohjelmistorobotiikan asiakkaat eivät ole kovinkaan teknisiä henkilöitä, joten tarkkaa teknistä prosessin kuvausta ei heiltä voi olettaakaan. Tämä ilmeni aluksi hankaluutena saada asiakkaat kuvaamaan automatisoitava prosessi riittävän selkeästi.

”Business people and developers must work together daily throughout the project.”

Ohjelmistorobotiikan soveltamisprojektien aloituspalaverista lähtien on oleellista olla jatkuvasti yhteydessä liiketoiminnasta vastaaviin henkilöihin eli asiakkaisiin, jotta ohjelmistokehitys etenisi haluttuun suuntaan ja vaatimusten mukaisesti. Tätä yhteydenpitoa varten Patriassa perustettiin Microsoft Teams-alustalle ohjelmistorobottikohtaisia ryhmiä, joissa on mukana ohjelmistokehittäjän lisäksi vain ne henkilöt, joita kyseinen ohjelmistorobotti

koskee. Teams -kanavan kautta voidaan välittää tiedostoja, mikä nopeuttaa tiedon liikkumista huomattavasti. Erääksi ohjelmistorobottien kehittämisen merkittäväksi pullonkaulaksi todettiin sen kehitykseen osallistuvien osapuolten tavoittaminen. Teams -kanavan avulla tiedonvaihto nopeutui huomattavasti ja projekteja voitiin viedä eteenpäin sujuvammin. Tämä ei kuitenkaan poista palaverien tärkeyttä.

”The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.”

Projektit aloitetaan aina kasvotusten käytävällä palaverilla, jossa automatisoitava prosessi arvioidaan ja sen kulku käydään läpi. Aloituspalaverissa on keskeistä arvioida prosessin automatisoinnin kannattavuus ja vaativuus. Mikäli kyseinen prosessi todetaan kannattavaksi automatisointikohteeksi, tulee seuraavaksi tunnistaa prosessin kulku pääpiirteissään. Oleellista on tunnistaa prosessin keskeisimmät vaiheet ja aloittaa ohjelmistorobotin kehittäminen niistä. Kun automatisoitavan prosessin keskeisimmät vaiheet on toteutettu, voidaan sitä laajentaa loogisesti.

“The best architectures, requirements, and designs emerge from self-organizing teams.”

Tämä nousi tässä vaiheessa erittäin keskeiseksi tekijäksi Patrian ympäristössä johtuen siitä, että mitään valmiita määriteltyjä sovellusarkkitehtuureja tai malleja ohjelmistorobotiikalle ei ollut vaan ohjelmistojen rakenteet muuttuivat projektista toiseen. Tärkeää onkin ollut tunnistaa keskeisimmät periaatteet, joilla ohjelmistorobottien ohjelmien hierarkioita määritellään järkeviksi.

Näiden edellä esitettyjen näkökulmien lisäksi ohjelmistorobottien todettiin tehostavan prosessien suorituskykyä. Molemmissa tapaustutkimuksissa prosessien suorituskyky parantui huomattavasti. Ensimmäisessä tapaustutkimuksessa toteutetun ohjelmistorobotin todettiin lyhentävän laskujen läpimenoaikaa puolella Tämä on vuosi tasolla merkittävä säästö, koska laskujen määrät ovat useita tuhansia. Tapauksessa ROB0002 viikoittaisen työajan säästön todettiin olevan neljä tuntia. Molemmissa tapauksissa on syytä huomata, että suorituskyvyn lisäksi ohjelmistoroboteilla oli myönteinen vaikutus työntekijöihin. He huomasivat, ettei heidän jatkossa tarvitse tehdä yksitoikkoiseksi koettua työtehtävää vaan he voivat hyödyntää asiantuntijuuttaan ja keskittyä tulosten tulkitsemiseen sekä mahdollisten poikkeamien juurisyiden selvittämiseen.

7. YHTEENVETO

Tässä tutkimuksessa ohjelmistorobotiikalla pyrittiin yrityksen sisäisten prosessien tehokkuuden parantamiseen. Kyseisten prosessien tarkastelu ei ollut tutkimuksen keskiössä vaan tutkimuksessa keskityttiin itse ohjelmointirobottien kehitys- ja käyttöönottoprosessiin. Kuten tutkimuksen kirjallisuusselvityksessä todettiin, on ohjelmistorobotiikan soveltamisen kulmakivenä prosessien soveltuvuuden arviointi, jonka perusteella päätös automatisoinnista tehdään. Näitä tunnistettuja kriteereitä ohjelmistorobotiikalla automatisoitaville prosesseille sovellettiin tapaustutkimuksissa. Tapaustutkimusten perusteella saatiin näyttöä ohjelmistorobotiikan soveltuvuudesta Patrian prosessien automatisointiin. Tutkimustulosten mukaan ketterät menetelmät soveltuvat hyvin ohjelmistorobottien kehittämiseen ja käyttöönottoon.

Lisääntyvän kokemuksen sekä osaamisen myötä automatisointiprojektien toteuttaminen nopeutuu, koska kaikkea ei tarvitse enää tehdä alusta alkaen vaan esimerkiksi aiempien projektien ohjelmakoodeja voi hyödyntää uusissa projekteissa. Tämä on syytä ottaa huomioon ohjelmistokehitysvaiheessa, jotta usein esiintyviä toiminnallisuuksia kuten tietojärjestelmään kirjautumisia voidaan tehokkaasti hyödyntää ja näin säästää aikaa kehitystyön osalta. Toiseksi ohjelmistorobotiikka pakottaa yrityksen sisäisiä prosesseja tiettyyn muottiin, prosesseista tehdään yhdenmukaisia samalla kun niitä automatisoidaan. Tämä taas johtaa yrityksen operatiivisen tehokkuuden parantumiseen. Toisaalta on syytä huomata, että yrityksen panostaessa liikaa sisäisen tehokkuuden kehittämiseen saattaa panostus strategian kehittämiseen jäädä vähemmälle huomiolle. Yrityksen onkin syytä pitää mielessä jatkuva strategian päivittäminen ja kehittäminen, jotta se erottuisi edukseen kilpailijoistaan sekä pysyisi kilpailukykyisenä.

On syytä muistaa, että ohjelmistorobotit eivät ole yhtä luotettavia kuin järjestelmäintegraatioon perustuva automatisointi. Näin ollen ohjelmistorobotiikan näkökulmasta taustalla toimivien tietojärjestelmien kehittämistä ei pidä unohtaa ohjelmistorobotteja sovellettaessa. Koska ohjelmistorobotit rakennetaan näiden tietojärjestelmien päälle saattaa niiden ylläpidosta ja päivittämisestä muodostua työläs prosessi taustalla olevan tietojärjestelmän päivittyessä. Tyypillisesti tietojärjestelmän käyttäjälle ilmoitetaan järjestelmän tulevasta päivityksestä tai huoltokatkosta näyttämällä esimerkiksi ponnahdusikkuna järjestelmä avattaessa. Koska tällainen tilanne ei ole tyypillinen, ei ohjelmistorobotti kykene siihen varautumaan eikä suorittamaan ohjelmaansa normaalisti. Näin ollen oleellista on tiedonvälitys järjestelmän päivityksistä vastaavan tahon ja ohjelmistorobottien ylläpitäjän välillä, jotta tieto tulevasta muutoksesta saadaan ennen kuin ohjelmistorobotti yrittää suorittaa ohjelmansa.

7.1 Jatkotutkimusehdotukset

Ohjelmistorobotiikan soveltaminen Patrian tietohallinnossa on vasta alussa ja kuten aiemmin mainittiin, on tämän diplomityön tarkoitus luoda pohja ohjelmistorobotiikan osaamiskeskukselle. Mitä enemmän ohjelmistorobotteja tehdään ja käyttöönotetaan, sitä enemmän sen kehitykseen osallistuvilla henkilöillä tulee olla pääsyoikeuksia yrityksen eri järjestelmiin. Yrityksen rahaliikenteeseen liittyvissä prosesseissa on aina mukana useita eri henkilöitä erilaisissa rooleissa. Prosessien arvioinnissa mainittiin vaarallisen työn yhdistelmä, jolla tarkoitetaan tilannetta, jossa yhdellä henkilöllä on mahdollisuus hyväksyä yritykseen tuleva lasku ja maksaa se. Tulevaisuudessa laajennettaessa ohjelmistorobotiikan soveltamista tämä tulee ottaa huomioon arvioitaessa uusia potentiaalisia automatisoitavia prosesseja. Ohjelmistorobotin näkökulmasta vaarallisen työn yhdistelmällä ei ole merkitystä, koska se vain suorittaa sille määriteltyjä toimintoja. Ohjelmistorobotista kuitenkin vastaa aina jokin henkilö, jolla on ohjelmistorobotin kautta pääsyoikeudet kaikkiin sen operoimiin järjestelmiin ja prosesseihin. Tämä tulee ottaa huomioon tarkasteltaessa uusia potentiaalisia automatisoitavia prosesseja. Puhuttaessa vaarallisista työyhdistelmistä tarkoitetaan myös tilanteita, joissa henkilö voi sekä pyytää, että myöntää pääsyoikeudet itsellensä.

Kuten muissakin ohjelmistokehityshankkeissa viimeinen vaihe on järjestelmän ylläpito. Ylläpito rajattiin tämän tutkimuksen ulkopuolelle, mutta se on erittäin oleellinen seikka ottaa huomioon ohjelmistorobotiikkaa sovellettaessa. Mitä enemmän ohjelmistorobotteja organisaatiolla on käytössä, sitä suurempi on niiden vaatima ylläpidon määrä. Ylläpidon keskeisessä roolissa on ohjelmistopäivitykset ja havaittujen virheiden korjaamiset. Kuten aiemmin mainittiin, perustuu ohjelmistorobotiikka pitkälti käyttöliittymän läpi suoritettavaan ohjelmointiin. Muutokset tietojärjestelmissä, joita ohjelmistorobotit operoivat, aiheuttavat ohjelmistorobottien toiminnan kannalta ongelmia. Erityisen herkkiä tietojärjestelmäpäivityksille ovat ohjelmistorobotit, joiden ohjelmointi perustuu kuvan tunnistukseen. Tästä syytä ylläpidon kannalta on erityisen tärkeää käydä sujuvaa vuoropuhelua ohjelmistopäivityksistä vastaavien tahojen kanssa, jotta muutokset ohjelmistorobotin toimintaan osataan toteuttaa.

7.2 Tulevaisuuden suuntaviivoja

Yritykset eivät toimi tyhjiössä, vaan ne ovat jatkuvassa vuorovaikutuksessa ympäristönsä kanssa. Myöskään teknologinen kehitys ei tapahdu tyhjiössä ja siihen vaikuttavat monet tekijät. Hyvin karkealla tasolla teknologian kehittymiseen vaikuttavat voimat voidaan jakaa sisäisiin ja ulkoisiin. Sisäisinä vaikutuksina ajatella itse teknologian kehittyminen. Tässä pätee teknologian imperatiivi, jonka mukaan kaikki mitä voidaan kehittää. Ulkoisina vaikutuksina voidaan ajatella ei-tekniset vaikutukset kuten lainsäädäntö, kysyntä tai sosiaaliset trendit. Tulevaisuuden ennustaminen on usein hankalaa, niin myös teknologioiden kohdalla, mutta nykyisiä trendejä tarkastelemalla voidaan hahmotella jonkinlaista

kuvaa mahdollisista kehityssuunnista [22]. Ohjelmistorobotiikkaa tulisi tarkastella siirtymävaiheen teknologiana kohti entistä automaattisemmin toimivia järjestelmiä. Käyttöliittymän läpi tapahtuvassa prosessien automatisoinnissa ei ole itsessään ole juurikaan mitään uutta, mutta ohjelmistorobottien käyttäminen on saavuttamassa yleisen hyväksynnän varteenotettavana sovellusteknologiana. Uuden teknologian käyttöönottaminen edellyttää riittävän kypsää teknologiaa ja kyseisen teknologian sosiaalista hyväksymistä. Uusiin teknologioihin liittyy aina ennakkoluuloja. Puhuttaessa tekoälystä ja algoritmien yleistymisestä pelkona on ollut työpaikkojen katoaminen ja ihmisten tuleminen tietokoneiden syrjäyttämiksi. Pelot tekoälyn yleistymisestä olisikin syytä kääntää kysymykseksi, miten toimimme sen kanssa ja mitkä tehtävät haluamme ulkoistaa tietokoneen tekemiksi.

Tällä hetkellä niin ohjelmistorobotiikassa kuin koneoppimisessakin kyse on hyvin pitkälti rutiininomaisten tehtävien automatisoinnissa. Kehittyvä teknologia mahdollistaa ihmisten työajan vapauttamisen sellaisiin tehtäviin, joissa tarvitaan ihmisen päättelykykyä ja harkintaa. Tällaiseen ihmisaivoille ominaiseen kognitiiviseen toimintaan tietokoneet eivät kykene, ainakaan vielä. Teknologinen kehitys on edennyt viime vuosina huimaa vauhtia, eikä kehityksen hidastumista ole näköpiirissä. Nähtäväksi jääkin mitä kaikkea teknologinen kehitys tulevaisuudessa mahdollistaa.

LÄHTEET

[1] Aleksandre Asatiani, Esko Penttinen Aalto University School of Business, Helsinki, Finland Turning robotic process automation into commercial success – Case OpusCapita

[2] Anagnoste, S. (2018). Setting up a robotic process automation center of excellence. Management Dynamics in the Knowledge Economy, 6(2), 307-322.

[3] DeBrusk, Chris. Weblog post. MIT Sloan Blogs, Five Robotic Process Automation Risks to Avoid, Cambridge: Massachusetts Institute of Technology, Cambridge, MA. Oct 24, 2017.

[4] Raili Saarinen, Birgitta Selonen, Leena Olkkonen, Patrialainen, Patrian henkilöstölehti, 6/2018

[5] Lacity, Mary C., Willcocks, Leslie, Robotic Process Automation at Telefónica O2. MIS Quarterly Executive. Mar 2016, Vol. 15 Issue 1, p21-35. 15p

[6] Hevner, A. R., March, S. T., Park, J., & Ram, S. (2008). Design science in information systems research. *Management Information Systems Quarterly*, *28*(1), 6.

[7] John Wiley & Sons, Ltd. 2011. Agile Software Development. Published online Wiley Online Library

[8] Coleman, G. (2016). Agile software development. Software Quality Professional, 19(1), 23-29. Retrieved from https://search-proquest-com.libproxy.tut.fi/docview/1866513328?accountid=27303

[9] John Wiley & Sons, Ltd. 2010. Measuring the flow in lean software development. Published online in Wiley Online Library

[10] Holcombe, Mike. Running an Agile Software Development Project, John Wiley & Sons, Incorporated, 2008. ProQuest Ebook Central, http://ebookcentral.proquest.com/lib/tut/detail.action?docID=427745.

[11] Capgemini Consulting: Robotic Process Automation-Robots conquer business processes in back offices (2016).

[12] Lacity, M., Willcoks, L.: What knowledge workers stand to gain from automation. Harvard Bus. Rev. (2015).

[13] Fung, H.P.: Criteria, use cases and effects of information technology process automation (ITPA). Adv. Robot. Autom. 3, 1–11 (2014)

[14] Tore Dybå, Torgeir Dingsøyr, Empirical studies of agile software development: A systematic review, Information and Software Technology, Volume 50, Issues 9–10, 2008, Pages 833-859, ISSN 0950-5849

[15] J. Erickson, K. Lyytinen, K. Siau, Agile Modeling, Agile software development, and extreme programming: the state of research, Journal of Database Management 16 (4) (2005) 88–100.

[16] https://www.patria.fi/fi/patria/konsernirakenne, viitattu 20.9.2018

[17] Aguirre S., Rodriguez A. (2017) Automation of a Business Process Using Robotic Process Automation (RPA): A Case Study. In: Figueroa-García J., López-Santana E., Villa-Ramírez J., Ferro-Escobar R. (eds) Applied Computer Sciences in Engineering. WEA 2017. Communications in Computer and Information Science, vol 742. Springer, Cham

[18] Dima, A. M., & Maassen, M. A. (2018). From Waterfall to Agile software: Development models in the IT sector, 2006 to 2018. Impacts on company management. Journal of International Studies, 11(2), 315-326. doi:10.14254/2071-8330.2018/11-2/21

[19] Youssef Bassil (2012). A Simulation Model for the Waterfall Software Development Life Cycle. International Journal of Engineering & Technology (iJET), ISSN: 2049-3444, Vol. 2, No. 5, 2012

[20] Sandro Javier Bolaños Castro, Rubén Gonzalez Crespo, Victor Hugo Medina Garcia (2011). Patterns of Software Development Process International Journal of Interactive Multimedia and Artificial Intelligence. 2011;1(4):33-40

[21] Shigeru YAMADA, Toshiki AOKI (2008). Quantitative Software Quality/Reliability Prediction Based on Project Management Data for Waterfall and Agile Development Paradigms. OPSEARCH. Vol. 45, No.4, 2008

[22] Martin G. Moehrle, Ralf Isenmann, and Robert Phaal (2012). Technology Roadmapping for Strategy and Innovation.

[23] DAVID COHEN, MIKAEL LINDVALL, AND PATRICIA COSTA (2004). An Introduction to Agile Methods. Volume 62, 2004, Pages 1-66

[24] Jon-Arild Johannessen. Automation, Innovation and Economic Crisis: Surviving the Fourth Industrial Revolution by PUBLISHER Routledge DATE 2018-05-15

[25] Philippe Dugerdil and Mihnea Niculescu, Visualizing Software Structure Understandability (2014). Geneva School of Business Administration

[26] Areej Sewalh AL_Zaidi, M. Rizwan Jameel QureshiScrum (2014). Practices and Global Software Development. I.J. Information Engineering and Electronic Business, 2014, 5, 22-28 Published Online October 2014 in MECS

[27] Sarah CroweEmail author, Kathrin Cresswell, Ann Robertson, Guro Huby, Anthony Avery and Aziz Sheikh (2011). The case study approach. BMC Medical Research Methodology 2011

[28] Outsourcing insider: Adapting to robotic process automation (2017). Chatham: Newstex. Retrieved from https://search-proquest-com.libproxy.tut.fi/docview/1872655333?accountid=27303

[29] Stephen Denning, (2015) "Updating the Agile Manifesto", Strategy & Leadership, Vol. 43 Issue: 5, https://doi-org.libproxy.tut.fi/10.1108/SL-07-2015-0058

[30] Kruchten, Philippe. Contextualizing agile software development. Journal of software: Evolution and Process, 04/2013, Vuosikerta/volyymi 25, Numero 4

[31] Petersen, K, Wohlin C. 2009. A comparison of issues and advantages in agile and incremental development between state of the art and industrial case. Journal of Systems and Software, 82(2009)9, pp. 1479-1490

[32] Sharp, Alec. Workflow Modeling: Tools for Process Improvement and Application Development, edited by Patrick McDermott, Artech House, 2008. ProQuest Ebook Central, https://ebookcentral.proquest.com/lib/tut/detail.action?docID=456899.

[33] Harrison, K. B., & Wanyama, P. (2001). Automation and its impact on the job satisfaction among the staff of the margaret thatcher library, moi university. *Library Management, 22*(6), 303-310. doi: http://dx.doi.org.libproxy.tut.fi/10.1108/EUM0000000005597

[34] Fassoulis, K., & Alexopoulos, N. (2015). The workplace as a factor of job satisfaction and productivity. *Journal of Facilities Management, 13*(4), 332-349. Retrieved from https://search-proquest-com.libproxy.tut.fi/docview/1705411075?accountid=27303

[35] N. Modig, P Åhlström, Tätä on Lean, Rheologica Publishing, Ruotsi 2016, s.29

[36] K. Laamanen, M. Tinnilä, Prosessijohtamisen käsitteet, Teknologiainfo Teknova Oy, Suomi, 2009